Development of
An MR Image Manipulation Tool
Utilising Relational Database Management System
(RDBMS)

by

H. R. Biswas

A Thesis Submitted
to the School of Electronics and Physical Science
at the University of Surrey
for the Degree of Master of Philosophy

University of Surrey

July 2005
Abstract

This thesis is the result of an MPhil study at the University of Surrey utilising the MRI dataset provided by The Institute of Cancer Research and Royal Marsden Hospital NHS Trust. The objective was to investigate and develop an MR Image manipulation tool within a relational database management system (RDBMS) environment.

In this study the main imaging modalities, such as x-ray computed tomography (CT), emission computed tomography, magnetic resonance imaging (MRI), ultrasound, biomagnetic source imaging and other digital medical imaging techniques, are reviewed. The principle of magnetic resonance imaging is examined to understand how an MR image manipulation tool within RDBMS can be useful for quantitative MR image analysis for diagnostic medical research such as cancer.

Currently thousands of MR images might produced for a single cancer patient during the course of his treatment, but there is a lack of effective computer aided systems for managing and analysing these images, and for assisting clinicians in their decision making process. This thesis identifies the development of an effective MR image manipulation tool within relational database management system environment as a possible resolution of this problem.

Detailed specifications for designing an MR image database tool are researched and the generalised standard is proposed in this thesis. The proposed MR image database tool mainly covers 4 areas: (1) definition of database for storing MR images within RDBMS, (2) definition of import routine for processing and importing raw MR images into the database, (3) definition of MR image search and selection tool for querying images from the database and (4) definition of standard for developing MR image plug-in module such
as the example program for calculating $T_2^*$ and $M_0$ images utilising images from MR image database.

A prototype of MR image database tool is designed and developed to realise the above four main functionalities and its capability is demonstrated within this research work. Though the image import routine may vary for MR images acquired from different MRI scanners, the underlying database storage, the search engine and the plug-in module remain the same. This adds enormous benefits toward the diagnostic medical imaging research work.
“Employ your time in improving yourself by other men’s writings so that you shall come easily by what others have laboured hard for”

Socrates

(469-399 BC.)
Acknowledgements

I would like to gratefully acknowledge my supervisor at the University of Surrey, Dr. Simon J. Doran, for his invaluable help and guidance throughout the project work. As the pressure mounted very high from my full-time job, and there were moments when I felt very down and wanting to give up this project work, Dr Doran was always very kind and patient with me, and helped me to stay on course. I am really grateful to him, thank you.

I would like to acknowledge my gratitude toward David Collins of the Institute of Cancer Research, Royal Marsden Hospital NHS Trust for supporting me with all necessary experimental MR image data.

I would like to thank UltraSoft Technologies Ltd for providing me with the necessary fund and time to carry out this research work.

I would like to thank my parents for their endless prayers for me. I am really lucky to have them.

But most of all I would like to thank my wife Micheline for her moral support; and my children Yasmin, Nafiz, Yasin and Sarah for giving up much of their bedtime-story time and play time over the last 3 years. Thank you all, I could not achieve this without your understanding.
## Abbreviations and Symbols

The following lists contain abbreviations and symbols that are most frequently used in this thesis.

**List of abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>two dimension</td>
</tr>
<tr>
<td>3D</td>
<td>three dimension</td>
</tr>
<tr>
<td>4D</td>
<td>four dimension (or real-time 3D), time as the forth dimension</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>CSF</td>
<td>cerebrospinal fluid</td>
</tr>
<tr>
<td>CR</td>
<td>computed radiography</td>
</tr>
<tr>
<td>CT</td>
<td>computed tomography</td>
</tr>
<tr>
<td>DB</td>
<td>database</td>
</tr>
<tr>
<td>DF</td>
<td>digital fluoroscopy</td>
</tr>
<tr>
<td>DSA</td>
<td>digital subtraction angiography</td>
</tr>
<tr>
<td>DSC</td>
<td>dynamic susceptibility contrast</td>
</tr>
<tr>
<td>DWI</td>
<td>diffusion weighted imaging</td>
</tr>
<tr>
<td>ECT</td>
<td>emission computed tomography</td>
</tr>
<tr>
<td>EEG</td>
<td>electroencephalography</td>
</tr>
<tr>
<td>EPI</td>
<td>echo planar imaging</td>
</tr>
<tr>
<td>ETL</td>
<td>echo train length</td>
</tr>
<tr>
<td>FSE</td>
<td>fast spin echo</td>
</tr>
<tr>
<td>GRASE</td>
<td>gradient and spin-echo pulse sequence</td>
</tr>
<tr>
<td>GRE</td>
<td>gradient-recalled echo</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>JDBC</td>
<td>java database connectivity</td>
</tr>
<tr>
<td>MEG</td>
<td>magnetoencephalography</td>
</tr>
<tr>
<td>MR</td>
<td>magnetic resonance</td>
</tr>
<tr>
<td>MRA</td>
<td>magnetic resonance angiography</td>
</tr>
<tr>
<td>MRI</td>
<td>magnetic resonance imaging</td>
</tr>
<tr>
<td>MTC</td>
<td>magnetization transfer contrast</td>
</tr>
<tr>
<td>MTR</td>
<td>magnetization transfer ratio</td>
</tr>
<tr>
<td>NetBEUI</td>
<td>NetBIOS enhanced user interface</td>
</tr>
<tr>
<td>NetBIOS</td>
<td>network basic input/output system</td>
</tr>
<tr>
<td>NM</td>
<td>nuclear medicine</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>NMR</td>
<td>nuclear magnetic resonance</td>
</tr>
<tr>
<td>NHS</td>
<td>national health service (United Kingdom)</td>
</tr>
<tr>
<td>ODBC</td>
<td>open database connectivity</td>
</tr>
<tr>
<td>PACS</td>
<td>picture archiving and communication system</td>
</tr>
<tr>
<td>PC</td>
<td>phase-contrast</td>
</tr>
<tr>
<td>PD</td>
<td>proton density</td>
</tr>
<tr>
<td>PET</td>
<td>positron emission tomography</td>
</tr>
<tr>
<td>RARE</td>
<td>rapid acquisition with relaxation enhancement</td>
</tr>
<tr>
<td>rCBF</td>
<td>relative cerebral blood flow</td>
</tr>
<tr>
<td>rCBV</td>
<td>relative cerebral blood volume</td>
</tr>
<tr>
<td>RDBMS</td>
<td>relational database management system</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio</td>
</tr>
<tr>
<td>SPECT</td>
<td>single photon emission computed tomography</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>transmission control protocol/internet protocol</td>
</tr>
<tr>
<td>TOF</td>
<td>time-of-flight</td>
</tr>
<tr>
<td>TR</td>
<td>time of repetition</td>
</tr>
<tr>
<td>US</td>
<td>ultrasound</td>
</tr>
</tbody>
</table>
List of symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>magnetic field</td>
</tr>
<tr>
<td>$B_0$</td>
<td>stationary magnetic field</td>
</tr>
<tr>
<td>G</td>
<td>magnetic field gradient</td>
</tr>
<tr>
<td>$G_x$</td>
<td>read(out) gradient (magnitude)</td>
</tr>
<tr>
<td>$G_y$</td>
<td>phase-encoding gradient (magnitude)</td>
</tr>
<tr>
<td>$G_z$</td>
<td>slice-selective gradient (magnitude)</td>
</tr>
<tr>
<td>Hz</td>
<td>hertz (unit of frequency)</td>
</tr>
<tr>
<td>I</td>
<td>angular moment</td>
</tr>
<tr>
<td>k</td>
<td>k-space vector,</td>
</tr>
<tr>
<td>KHz</td>
<td>kilohertz (=1000 Hz)</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>gyro-magnetic ratio</td>
</tr>
<tr>
<td>$\mu$</td>
<td>dipole magnetic moment</td>
</tr>
<tr>
<td>M</td>
<td>total magnetization</td>
</tr>
<tr>
<td>$M_0$</td>
<td>equilibrium magnetization</td>
</tr>
<tr>
<td>$M_{xy}$</td>
<td>transverse magnetization</td>
</tr>
<tr>
<td>t</td>
<td>time</td>
</tr>
<tr>
<td>$\tau$</td>
<td>pulse (echo) interspacing in burst</td>
</tr>
<tr>
<td>T</td>
<td>tesla</td>
</tr>
<tr>
<td>$T_0$</td>
<td>delay between excitation and acquisition</td>
</tr>
<tr>
<td>$T_1$</td>
<td>longitudinal (spin-lattice) relaxation time</td>
</tr>
<tr>
<td>$T_2$</td>
<td>transverse (spin-spin) relaxation time</td>
</tr>
<tr>
<td>$T_2^*$</td>
<td>$T_2$ star</td>
</tr>
<tr>
<td>$T_A$</td>
<td>image acquisition time</td>
</tr>
<tr>
<td>$T_E$</td>
<td>time to echo</td>
</tr>
<tr>
<td>$T_R$</td>
<td>repetition time</td>
</tr>
<tr>
<td>S(k)</td>
<td>NMR signal in k-space</td>
</tr>
<tr>
<td>S($k_x,k_y$)</td>
<td>measured k-space signal (raw data)</td>
</tr>
<tr>
<td>S(x,y)</td>
<td>MR image</td>
</tr>
<tr>
<td>$\omega$</td>
<td>larmor frequency</td>
</tr>
<tr>
<td>$\rho$</td>
<td>proton density</td>
</tr>
</tbody>
</table>
\( \Delta R_2^* \) \( T_2^* \) relaxation
\( \hat{x}, \hat{y}, \hat{z} \) unitary vectors of laboratory frame of reference
\( \hat{x}', \hat{y}', \hat{z}' \) unitary vectors of rotating frame of reference
# Table of Contents

Abstract .......................................................................................................................... iii

Acknowledgements .......................................................................................................... vii

Abbreviations and Symbols ........................................................................................... viii

CHAPTER - 1 .................................................................................................................. 1

1 Introduction .................................................................................................................. 1

1.1 What is medical imaging? ......................................................................................... 1

1.2 What can you find out from medical imaging? ........................................................ 2

1.3 Main Imaging modalities .......................................................................................... 3

1.3.1 X-Ray Computed Tomography (CT) ................................................................. 3

1.3.2 Emission Computed Tomography ......................................................................... 7

1.3.3 Magnetic Resonance Imaging ............................................................................. 8

1.3.4 Ultrasound ......................................................................................................... 9

1.3.5 Biomagnetic Source Imaging ............................................................................. 10

1.3.6 Digital extensions of plane-film x-ray ............................................................... 11

1.3.7 Other digital medical imaging ........................................................................... 11

1.4 Relational database management system ............................................................... 12

1.4.1 Introduction ....................................................................................................... 12

1.4.2 Database in commercial system .......................................................................... 13

1.4.3 Database in medical imaging ............................................................................ 14

1.4.4 Quantitative imaging using database tools ....................................................... 15

1.5 Aim of the project ................................................................................................... 16

CHAPTER - 2 .................................................................................................................. 17

2 Magnetic Resonance Imaging ......................................................................................... 17

2.1 Fundamental physics of MRI .................................................................................. 17

2.1.1 NMR phenomenon ......................................................................................... 17

2.1.2 RF excitation .................................................................................................... 19

2.1.3 Relaxation ......................................................................................................... 20

2.1.4 Conclusion ....................................................................................................... 21

2.2 Spatial encoding in MRI ........................................................................................ 22

2.2.1 Selective excitation ......................................................................................... 23
1.1 What is medical imaging?

Different definitions exist for “medical imaging” [2]. According to Schmitz [3] “medical imaging” is a scientific process for creating images of the interior of the living human body from the outside for diagnostic purposes. Again, the NHS information authority defines medical images as “all images used in the diagnosis and treatment of patients” [5]. This can include any form of image, such as X-ray films, photographs of histological specimens, Computed Tomography (CT) images, Ultrasound scans, MR images and even photographs of the patient used for identification purposes.

Medical imaging involves multi-disciplinary science, which can be summarised as following [1,3]:

- Physics (matter, energy, radiation, magnetism, etc.)
- Mathematics (linear algebra, calculus, statistics)
- Biology/Physiology
- Engineering (research, development and implementation)
- Computer science (image reconstruction, storage & manipulation, signal processing)

Diagnostic medical imaging mainly utilises magnetic resonance, electromagnetic radiation, ultrasonography or radioactivity for evaluation of body tissues in order to diagnose injury and disease by means of radiological images [1].
1.2 What can you find out from medical imaging?

Medical imaging is the most important source of anatomical and functional information, which is indispensable for today's clinical research, diagnosis and treatment, and is an integral part of modern health care [4]. Medical images are at the heart of the patient's diagnosis, surgical planning, therapy and long-term follow-up for outcome assessment [1].

Recent advances in medical imaging also enable medical professionals to detect many important diseases such as cancer at a very early stage, and take appropriate measures to treat or cure it.

However, the cost of many of these techniques and the limited number of facilities make it currently impractical to use them for mass screening, except for the case of ultrasound scanning during pregnancy, where the benefits are now clearly established.
1.3 Main imaging modalities

The introduction of advanced imaging technologies has significantly improved the quality of medical care available to patients. Non-invasive imaging modalities allow a physician to make increasingly accurate diagnoses and render precise and measured modes of treatment. Current uses of imaging technologies include laboratory medicine, surgery, radiation therapy, nuclear medicine, and diagnostic radiology. Common medical imaging methodologies may be divided grossly into two general groupings: (a) techniques that seek to image internal anatomical structures and (b) methods that present mappings of physiological function [6].

Rapid technological advances have made the acquisition of 3D and 4D [50] (time series 3D images) representations of human and animal internal structures commonplace. A multitude of imaging modalities are currently available.

1.3.1 X-Ray Computed Tomography (CT)

Computed Tomography (CT) is a radiographic technique that uses a computer to assimilate multiple X-ray images into a 2 dimensional cross-sectional image. This can reveal many soft tissue structures not shown by conventional radiography. Scans may also be dynamic in which a movement of a dye is tracked. Cuts (image slices) may be 5mm apart or, in some instances even further apart. A contrast agent may be injected into the patient’s vein prior to the scan to help differentiate abnormal tissue and vasculature. [51]
1 Introduction

Figure 0: Computed Tomography (CT) Equipment.

Computed tomography (CT) - sometimes called CAT scan, uses special x-ray equipment to obtain image data from different angles around the body then uses computer processing of the information to show a cross-section of body tissues and organs.

Courtesy: Radiology Info (http://www.radiologyresource.org/)

The scanning sensor of a typical CT scanner rotates 180° around the patient's body, sending out a pencil-thin X-ray or fan beam. Crystals positioned at the opposite points of the beam pick up and record the transmitted x-rays, thus allowing the absorption of the varying thickness of tissue and bone to be calculated. These data are then relayed to a computer that turns the information into a picture on a screen. Using radiation as that of the conventional X-ray machine, an entire slice of the body is made visible with full depth information, something that is not available on a traditional plane film x-ray. [51-52]

Since the first CT scanner was developed in 1972 by Sir Godfrey Hounsfield, the modality has become established as an essential radiological technique applicable in a wide range of clinical situations. CT scans are used to study many areas of the body and the limbs, including the following [54]:

**Chest (thorax):** A CT scan of the chest can detect infection, lung cancer, pulmonary embolism, and aneurysms. It can also be used to help determine whether cancer has spread (metastasized) into the chest from another location in the body.
1.3 Main imaging modalities

**Abdomen:** A CT scan of the abdomen can help detect several conditions, including cysts, abscesses, infection, tumors, an aneurysm, enlarged lymph nodes, foreign objects, bleeding into the abdominal cavity, diverticulitis, inflammatory bowel disease, and appendicitis. It can also help determine whether cancer has spread from another place in the body to abdominal organs or lymph nodes.

**Urinary tract:** A CT scan can detect kidney stones, blockage, abnormal growths, infection, structural problems, and some diseases of the urinary tract.

**Liver:** A CT scan can detect liver tumours, bleeding from the liver, and some liver diseases. It can also be used to determine whether cancer has spread from another part of the body. A CT scan of the liver can help determine the cause of jaundice.

**Pancreas:** A CT scan can detect a tumour in the pancreas or inflammation of the pancreas (pancreatitis).

**Gallbladder and bile ducts:** A CT scan can be used to investigate blockage of the bile ducts. Gallstones occasionally show up on a CT scan, but an ultrasound test is usually used to detect gallstones.

**Adrenal glands:** A CT scan can detect tumours or an enlargement of the adrenal glands.

**Spleen:** A CT scan can be used to evaluate injury to the spleen.

**Spine and spinal bones (vertebrae):** A CT scan can detect tumours, injuries, deformities, narrowing of the spinal canal (spinal stenosis), and other problems of the spine. The test can also identify a herniated disc of the spine and help determine whether complications of osteoporosis are present. For more information, see the medical test CT Scan of the Spine [54].
Advantages and disadvantages of CT [53]

**Advantages:**
- CT is readily available in most hospitals.
- It is an increasingly rapid imaging modality with excellent image resolution, enabling accurate diagnostic evaluation of patients over a wide spectrum of clinical indications.
- The data acquired in one scan can subsequently be manipulated to provide multiplanar and 3D reconstructions.

**Disadvantages:**
- Radiation – although CT scans account for only 4% of X-ray examinations, they contribute to more than 20% of the radiation dose to the population by 'medical X-rays'. [53]
- Artefacts – an artefact is a feature or appearance that is seen on an image, which does not actually exist. They occur in all imaging modalities and are often unavoidable. Recognizing the presence of artefacts is important in order to avoid confusion with pathology. However, with the increasing speed of image acquisition in a single breath hold by the most modern scanners, many artefacts are being minimized or eliminated. Types of artefacts include:
  1. *motion* – from patient movement during a scan, commonly due to breathing
  2. *streak* (beam hardening) – dark 'streaks' behind high-density objects, e.g. dental amalgam and metallic joint replacements
  3. *partial voluming* – different tissue densities within a single voxel lead to 'averaging' of data. For example, a small black object within a larger white space would look like a shade of grey.
- Relatively poor tissue contrast when compared to MRI can be a problem, despite the use of oral and IV contrast. This may occur in thin adults and children due to the lack of intra-abdominal fat separating the various tissue planes.
- CT has a relatively high cost and limited portability.
1.3 Main imaging modalities

- There can be contrast media-related complications, including allergic reactions and renal toxicity.

### 1.3.2 Emission Computed Tomography

Emission Computed Tomography (ECT) techniques obtain 3D representations of the location of injected pharmaceuticals. The injected pharmaceuticals are labelled with gamma ray emitting radionuclides. The emitted gamma rays are measured at sites external to a patient. Such measured data may be utilized to reconstruct a 3D mapping of internal emission density.

ECT differs fundamentally from many other medical imaging modalities in that it produces a mapping of physiological functions, such as perfusion and metabolism, as opposed to imaging anatomical structure [55].

In the case of **positron emission tomography (PET)** and **single photon emission (SPECT)** (which constitute the two different modalities of ECT), the radiation source utilised is a radioisotope compound that has been introduced into a patient’s body. The radioactive compound is metabolically distributed into various organs and tissues and no a priori information concerning the radiation’s location is known. The main goal of ECT is the determination of the isotope compound distribution within a patient’s body. Measurements are taken of radiation leaving the patient’s body using a specially designed detector system. The detector readings are employed as input to a reconstruction algorithm to produce the internal isotope compound distribution.

SPECT and PET involve the use of isotopes that emit high-energy particles like gamma rays and positrons (which result in the generation of gamma rays), respectively, to label probes [56 and 57]. Gamma-emitting isotopes include Technetium-99 m (Tc-99 m), Indium-111 (In-111), and Iodine-131, while the common positron emitting isotopes include Fluorine-18 (F-18), Carbon-11, Nitrogen-13, Iodine-124, and Copper-64 (Cu-64) (for reviews, see
To clarify, in PET, positrons are not directly detected. Positrons (particles with the same mass as an electron, but having the opposite charge) emitted from the atom travel up to a few millimetres from the source. Eventually, they undergo an annihilation reaction with an electron in the tissue resulting in the simultaneous emission of two gamma rays of 511 keV (the mass energy of an electron/positron) at approximately 180° apart. These two gamma rays are detected and used to determine the tomographic information collected in PET. SPECT also supplies tomographic information, by rotating position-sensitive "gamma camera" detectors around the specimen and only collecting gamma particles that hit the detectors through a collimator (an aperture that allows only particles travelling directly through the hole to be detected). Both of these modalities use only femtograms of radiolabeled tracer, offering high levels of sensitivity, and can be used with autoradiographic methods to correlate in vivo data [57].

Though SPECT is a technique similar to PET, the radioactive substances used in SPECT (Xenon-133, Technetium-99, Iodine-123) have longer decay times than those used in PET, and emit single instead of double gamma rays. SPECT can provide information about blood flow and the distribution of radioactive substances in the body. Also, SPECT centres are more accessible than PET centres because they do not have to be located near a particle accelerator [60].

1.3.3 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is similar to CT imaging in that it generates 3D data sets corresponding to a patient's anatomy. However, MRI differs fundamentally from CT in the manner in which images are acquired. CT scanners employ x-ray radiation to generate the data needed to reconstruct internal structures. By contrast, MRI utilises non-ionising RF waves and magnetic fields to obtain 3D images and is based on the principle of Nuclear Magnetic Resonance (NMR). MRI provides unparalleled soft tissue contrast in a non-invasive manner. For more detailed description, see chapter 2.
1.3.4 Ultrasound

Ultrasound images are obtained by using high-frequency sound waves that are transmitted through tissues and are then reflected back and detected. Diagnostic ultrasound using 7.5–15 MHz frequencies has a spatial resolution of 300–500 microns [61], while ultrahigh frequency (40 MHz) systems have been developed that are specifically used for mouse imaging that can obtain resolutions of 40–60 microns [62].

Typically, ultrasound is used to observe perfusion and anatomical characteristics. Ultrasound contrast agents, like lipid microbubbles (which affect the acoustic signal that is detected), have been developed that can target specific disease states by attaching proteins like albumin and antibodies to the microbubbles. Furthermore, microbubbles can be constructed to carry genes or therapeutic agents. Using ultrasound, the microbubbles can be destroyed at a specific site of disease to deliver the contents to the targeted tissues [63 and 64]. Review of ultrasound contrast agents can be found elsewhere [65].

Ultrasound based imaging techniques comprise a set of methodologies capable of acquiring both quantitative and qualitative diagnostic information. Many ultrasound based methods are attractive due to their ability to obtain real-time images, employing compact and mobile equipment, at a significantly lower cost than is incurred with other medical imaging modalities. The real-time nature of ultrasound makes it possible for physicians to observe the motion of structures inside a patient’s body. This ability has resulted in the widespread use of ultrasound technology in the fields of paediatrics and cardiology. Equipment which employs doppler echo techniques can extract quantitative velocity information such as the rate of blood flow in a vessel of interest. Additionally, the introduction of ultrasound signals into a patient, at the levels currently employed, has been determined to be safe [7]. The lack of negative effects from exposure, portability of equipment, relatively low cost
and quantitative acquisition modes distinguish ultrasound techniques as an important class of medical imaging technology.

### 1.3.5 Biomagnetic Source Imaging

Biomagnetic Source Imaging is another methodology, which gathers functional information. This technology allows for the external measurement of the low level magnetic fields generated by neuron activity. Biomagnetic source imaging allows a clinician to gather data concerning brain function that has previously proved elusive [6]. One of the most commercialised biomagnetic source imaging is Magnetoencephalography.

Magnetoencephalography (MEG) is a neurophysiological technique that allows the recording of magnetic fields generated by brain activity. The first recordings were performed by David Cohen in 1968, using a one-sensor magnetometer [66]. Up to the mid-eighties, MEG equipment was limited to one magnetic sensor or a few sensors, which precluded appropriate coverage of the brain and implied very prolonged acquisition times. Only in the last decade, when rapid development of MEG technology took place, did MEG devices with whole-scalp coverage with 150 or more sensors become available. Their distribution has increased worldwide in the last few years, and today, a few dozen laboratories worldwide use MEG for functional exploration of the human brain and are developing clinical applications of the method.

It is remarkable that, despite its more than 30 years of existence, MEG is still widely considered a new neurophysiological technique that is still searching for a place in routine neurological practice. This may seem quite surprising when considering that after electroencephalography (EEG), MEG is the only available neurophysiological technique able to trace neuronal activity with appropriate temporal resolution, i.e., in the millisecond range. This is in sharp contrast with other neuroimaging techniques widely used in neuroscience today such as PET, SPECT, and functional MRI, which measure neuronal activity indirectly and which have a rather poor temporal resolution ranging from seconds to several minutes. Nevertheless, the introduction of a pure
neurophysiological technique such as MEG has aroused enormous expectations in both the basic neuroscience and clinical neurology communities [67].

1.3.6 Digital extensions of plane-film x-ray

Because of modern advances in computer technology and its accessibility, computers are used to analyse plane-film x-rays. Here is a list of some digitised imaging techniques:

- Digitised X-ray (Film) - Digitisation of conventional X-ray film.
- Computed Radiography (CR) - Direct digitisation of the image created using X-rays, via large area solid state detectors.
- Digital Fluoroscopy (DF).
- Digital Subtraction Angiography (DSA).
- Bone densitometry – DEXA (dual-energy x-ray absorptiometry)
- Mammography
- Dentistry X-ray

For the purpose of clarity, please note the above imaging techniques excepts Digitised X-ray Film and Dentistry X-ray which are digitised x-ray films, are acquired in digital form.

1.3.7 Other digital medical imaging

Digital medical images which generally share the characteristics of one or more of the main modalities described earlier, are also becoming commonly available in the following areas:

- Endoscopy
- Microscopy
- Electron Microscopy
- Thermography
1.4 Relational database management system

1.4.1 Introduction

Relational Database Management System (RDBMS) is a type of database in which the database is organized and accessed according to the relationships between data values. The RDBMS was invented by a team lead by Dr. Edmund F. Codd and funded by IBM in the early 1970's [77]. The Relational Model of RDBMS is based on the principles of relational algebra. RDBMS has the ability to access data organised in tabular format (table) that can be related to each other by a common field (primary key field), and has the capability to recombine the data items from different tables, providing powerful tools such as SQL (structured query language) for fast data retrieval and data storage.

Relational Database Management Systems range from desktop databases like Filemaker Pro, FoxPro, Paradox and Microsoft Access to industrial-strength databases like IBM DB2, Informix, Microsoft SQL Server, Oracle, and Sybase SQL Server. There are a number of free open source databases of less industrial strength such as Borland InterBase, MySQL and PostGre are also available. Over last 15 years the rapid development of standard commercial RDBMS made it to be the integral part of almost all information management systems. Information systems that use proprietary databases may lock in the user community to the vendors of information systems, but those that use a commercial standard RDBMS allow users to port their data to a competitive superior tool.

Software applications that are designed with RDBMS, manage a structured collection of information, automatically maintaining defined data relationships, which can be accessed by simultaneous users to update or review data residing in the database. RDBMS does not only store data and manage data relationships, but also holds all data within one single data file which is known as a database device.
1.4.2 Database in commercial system

Most commercial enterprises in terms of their organisational activities, usage of information, flow of information, information security, data capture, data manipulation and data abstractions, relate real-world events and operations to the underlying database structures that support the overall organisations' information systems. It is almost inseparable any commercial business management system from its underlying relational database management system. In every stage of business process automation from marketing campaign and sales process to project manage and successful delivery of projects, business application together with RDBMS help organisations to automate business process, centralise information and provide with real-time analysis which are crucial for the survival of the business.

For example, all enterprise resource planning (ERP) systems such as Microsoft Great Plains, NetERP and SAP are founded on industrial strength RDBMS. A typical ERP system offers everything that an organisation needs within one powerful system solution. With ERP, organisations can manage financials, order fulfillment, purchasing, inventory, web presence, payroll and all other aspects of business. Most importantly, all the components are tightly integrated with one underlying RDBMS, giving users access to real-time business intelligence to make better decisions. This means business controllers can access comprehensive financial data based on up-to-the-minute bank balances, receivables and payables. Orders can also be filled more quickly and easily, with inventory levels updated in real time. And with full visibility into inventory, companies can lower inventory levels, analyze inventory costs and collaborate with vendors more effectively. Additionally, ERP applies time and billing to payroll with just minimum manual work, dramatically simplifying employee management.
1.4.3 Database in medical imaging

One of the first medical imaging database of brain-maps was developed by Bloom et al. [78]. Because the system was based upon a hypercard (Apple Computer Corp.) database, the functionality of the system was limited. Since then computer technologies and database management system evolved greatly and the initiatives for creating many medical imaging databases are undertaken. Here are a few medical imaging databases that hold a large number of images: [79]

- **Custom Medical Stock Photo (CMSP)** – is a part of of mediaMD.com, USA, and holds approximately 1,200,000 images.
- **Mediscan** – is an initiative from Medical-On-Line Ltd, UK, and holds approximately 1,000,000 images.
- **Medical Pathographic Library** – is a part of the Wellcome Trust, UK and hold around 160,000 images.
- **Science Photo Library (SPL)** – is compiled by Michael Marten, UK and holds around 100,000 images.

Most of these databases are commercial image databases and compiled from various medical domains, and are offering JPEG, GIF or DICOM\(^1\) format images of CT, MRI, radiography, ultrasound, endoscopy, microscopy, as well as pictures from biology, pharmacology, botany, drugs and medical treatments. But none of these image databases offers any quantitative image information or quantitative imaging tools.

---

\(^1\) DICOM stands for Digital Imaging and Communications in Medicine. The DICOM standard was initiated by a joint committee of the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) in 1983 to: (1) Promote communication of digital image information, regardless of device manufacturer (2) facilitate the development and expansion of picture archiving and communication systems (PACS) that can also interface with other systems of hospital information and (3) allow the creation of diagnostic information data bases that can be interrogated by a wide variety of devices distributed geographically. [80]
**1.4.4 Quantitative imaging using database tools**

Soon, medical imaging will not simply provide insight into the past or present state of a biological process — medical imaging will predict its future state. To a limited extent, this is possible already, but to enhance our abilities in the development of new imaging methods/probes are critical, as is the development of a quantitative groundwork for understanding the interrelationships of the physiologic parameters measured. To implement what is learned, a database management tool is needed.

Multi-dimensional image analysis is being used increasingly to arrive at surrogate\(^2\) end-points for drug development trials. Various imaging modalities such as computed tomography (CT), magnetic resonance imaging (MRI), positron emission tomography (PET) and ultrasound are used to analyse treatments for diseases such as cancer, multiple sclerosis, osteoarthritis, and Alzheimer's disease. However, extracting information from images can be tedious and is prone to high user variability. The medical image analysis community is moving towards advanced software systems specifically designed for drug development trials \[8\]. These systems can automatically identify the anatomy of interest in medical images (segmentation methods), can compare the anatomy over time or between patients (registration methods) and allow the quantitative extraction of anatomical features and the integration of the data and results into a database management system, automatically tracking the changes made to the data (audit trail generation).

---

\(^2\) A surrogate endpoint of a clinical trial is a laboratory measurement or a physical sign used as a substitute for a clinically meaningful endpoint that measures directly how a patient feels, functions or survives.
1.5 Aim of the project

The main objectives of this project are (a) to develop a framework for constructing an image database tool [chapter 3], and (b) to design and demonstrate a prototype MR image database manipulation tool [chapter 4] based on this framework to perform search and quantitative analysis on images.

Modern relational database management systems (RDBMS) are being used in all commercial environments for quantitative analysis of business related information, from storing simple dynamic web content to more complex financial information manipulation. In medical imaging world researchers are going ahead in creating many image manipulation tools in full speed, but most of the image data are still kept in normal file format, which requires the user to manage them manually. So searching raw images, manipulation of images, doing quantitative analysis and combining a particular result of analysis with other results, and so on still depend on manual process, which can be very time consuming. To a certain extent, it is impossible to manipulate and relate massive quantity of multi-set historic image data from a uncoordinated simple file format images, and it hinders the progress of the further research and development in imaging world. Therefore, through this project it is aimed to design and develop a RDBMS MR imaging tool, and devise protocols for the future development of add-in (plug-in) image manipulation tools to expand its capability, and finally demonstrate its user friendliness and usefulness.

In this thesis, chapter 3 analysis and specification of Framework for MRI Database Tool and chapter 4 design, develop and demonstration of MRI Database Tool prototype are my original research work.
2.1 Fundamental physics of MRI

CHAPTER - 2

2 Magnetic Resonance Imaging

Magnetic resonance imaging (MRI) represents a major innovation in medical imaging technology. MR imaging can provide detailed images of the human body with unparalleled soft tissue contrast in a non-invasive manner. The phenomenon of nuclear magnetic resonance (NMR) is now well established in the field of diagnostic medicine. The technique itself has existed for over 58 years in the field of physical chemistry. NMR was first discovered in 1946 by Bloch and Purcell, two researchers working independently at Stanford and Harvard, respectively. Both were awarded the Nobel Prize in 1952 for their work.

2.1 Fundamental physics of MRI

This study of the fundamental physics of MRI is based on several general MRI textbooks/articles [10-12, 15], as well as some on-line resources [13,14].

2.1.1 NMR phenomenon

Nuclear magnetic resonance (NMR) is a quantum mechanical phenomenon if considered on the level of single atoms, but classical mechanics can still be used to describe the total magnetization behaviour in a volume. This section is based on the classical mechanical interpretation. The quantum mechanical explanation of magnetic resonance (yielding, of course, the same general result) can be found in [13] and [12], pp. 397-399.

Most MRI is based on magnetic resonance in hydrogen nuclei, i.e., single protons. Consider a single proton with angular momentum \( I \) associated with its spin. The magnetic dipole moment associated with the proton is

\[ \mu = \gamma I \quad (1) \]
where $\gamma$ is known as *gyro-magnetic ratio* and takes a different value for each type of nucleus. If placed in a magnetic field $B$, the magnetic dipole will experience torque causing change in the angular momentum:

$$\frac{d\vec{A}}{dt} = \vec{\mu} \times \vec{B}$$  \hspace{1cm} (2)

Combining Eq. (1) and Eq. (2), we obtain the Larmor equation

$$\frac{d\vec{\mu}}{dt} = \gamma \vec{\mu} \times \vec{B}$$  \hspace{1cm} (3)

that describes precession of $\mu$ around $B$ with angular velocity

$$\omega = -\gamma B$$  \hspace{1cm} (4)

also called Larmor frequency (Fig. 1a). Note that the magnitude of the dipole magnetic moment remains the same. The total magnetic moment $M$ of a sample, also called magnetization, is a vector sum of magnetic moments of individual nuclei. It is easy to show that $M$ satisfies the Larmor equation (3) as well. In the absence of an external magnetic field, the individual magnetic moment orientation is random over the volume, and the total magnetization is zero. This changes if an external magnetic field $B_0 = B_y \hat{z}$ is applied to the sample. Spins align themselves either "up" or "down" i.e. with or against the field. A small fraction of dipoles aligns with the external field, creating a non-zero magnetization of the sample in the $z$-direction. Since individual dipoles precess out of phase around the $z$-axis, the transverse component $M_{xy}$ of the total magnetization is still zero.

![Figure 1: NMR principles. (a) Larmor precession; (b) Effective magnetic field in the rotating reference frame; (c) Magnetic resonance in the rotating reference frame; (d) Magnetic resonance in the laboratory reference frame.](image-url)
2.1.2 RF excitation

If a time-varying magnetic field $B_1$ (a radio frequency (rf) pulse) is applied in the plane perpendicular to $B_0$, the magnetization vector $M$ tilts away from the z-axis. Precession of $M$ as described by the Larmor equation (3) creates a variable electro-magnetic field that induces current in the receiver coil. The analysis of the behaviour of $M$ in the presence of such a field $B_1$ is much simpler in a rotating reference frame. Let's consider a reference frame rotating around the z-axis with angular velocity $\omega$. In this reference frame, the Larmor equation (3) becomes

$$\frac{dM}{dt} = \gamma M \times B - \omega M \times M,$$

where the additional term $- \omega M \times M$ represents the difference in motion of $M$ between the reference frame rotating at $\omega$ and the laboratory reference frame. Substituting $B = B_1 + B_0$, we obtain

$$\frac{dM}{dt} = \gamma M \times (B_1 + B_0 + \omega / \gamma),$$

i.e., in the rotating reference frame, the magnetic momentum $M$ precesses around the effective magnetic field (Fig. 1b)

$$B_e = B_1 + B_0 + \omega / \gamma = B_1 + (B_0 + \omega / \gamma)\hat{z}.$$

If we choose the reference frame such that $\omega = \omega$, then $B_1 = B\hat{x}'$ and the effective field becomes

$$B_e = B_1 \hat{x}' + (B_0 + \omega / \gamma)\hat{z}'$$

(Fig. 1b). Furthermore, if $\omega$ is equal to the Larmor frequency (4), then $B_0 = B\hat{z}'$ and the motion of vector $M$ in the rotating reference frame becomes rotation around the $x'$-axis (Fig. 1c). The amount of this rotation is called the flip angle. The angular velocity of that rotation is $\omega = \gamma B_0 = \gamma B_1$. In the laboratory reference frame, $M$ spirals into xy-plane, as shown in Fig 1d.

If the duration of the rf pulse is $\tau$, the flip angle is
2. Magnetic Resonance Imaging

\[ \alpha = \omega_1 \tau = \gamma B_1 \tau \]  

(10)

and the transverse magnetization immediately after the pulse is

\[ M_{xy} = M_0 \sin \alpha = M_0 \sin(\gamma B_1 \tau) \]  

(11)

where \( M_0 \) is the magnitude of the magnetization \( M \) at \( t=0 \). This transverse magnetization induces current in a receiver coil, producing the signal used in MRI. Flip angles of 90° are commonly used in MR experiments by setting \( \tau = \pi/(2\gamma B_1) \). For such pulses, all of the magnetization is flipped into transverse plane, achieving the strongest possible signal.

### 2.1.3 Relaxation

Two different types of relaxation are observed in magnetic resonance experiments: (1) longitudinal, or spin-lattice, relaxation and (2) transverse, or spin-spin, relaxation. The effects of relaxation processes are usually negligible during a short rf pulse, but they govern the magnetization behaviour in the stationary magnetic field after an rf pulse is switched off.

Spin-lattice relaxation is the process of the \( z \) component of \( M \) aligning over time with the stationary magnetic field \( B_0 \):

\[ M_z = M_0 (1 - e^{-t/T_1}) \]  

(12)

where \( T_1 \) is called the spin-lattice relaxation time, and \( M_0 \) is the equilibrium magnetization that depends on the proton density (PD) of the sample and the magnetic field strength \( B_0 \). Spin-spin relaxation is due to interactions between spins in the sample. It causes exponential relaxation in the transverse plane:

\[ \| M_{xy} \| = M_{xy0} e^{-t/T_2} \]  

(13)

where \( T_2 \) is called the spin-spin relaxation time, and \( M_{xy0} \) is the transverse magnetization immediately after the rf pulse, which is equal to \( M_0 \) for the 90° rf pulse.

Thus, the following equations describe precession of the magnetization vector \( M \) after a single 90° rf pulse:
2.1 Fundamental physics of MRI

\[ M_x(t) = M_0 (1 - e^{-i\gamma t}), \]
\[ M_z(t) = M_0 e^{-i\gamma t} \cos \omega t, \]
\[ M_y(t) = M_0 e^{-i\gamma t} \sin \omega t, \] \hspace{1cm} (14)

where \( \omega = \gamma B_0 \) is the Larmor frequency. The transverse magnetization is often described as a single complex number:

\[ M_{xy}(t) = M_0 e^{-i\gamma t} e^{i\omega t}. \] \hspace{1cm} (15)

If a sequence of 90° rf pulses with a period of TR (time of repetition) is used, then the longitudinal component of magnetization does not recover to the equilibrium value \( M_0 \) and Eq. (15) becomes

\[ M_{xy}(t) = M_0 (1 - e^{-\gamma t/2}) e^{-i\gamma t} e^{i\omega t}. \] \hspace{1cm} (16)

The time between the rf pulse and the time when the signal is measured in the receiver coil, is usually denoted \( T_e \) (time to echo). Ignoring the relaxation effects for the duration of the measurement, the transverse magnetization during the measurement is then

\[ M_{xy}(t) = M_0 (1 - e^{-\gamma t/2}) e^{-\gamma t/2} e^{i\omega t} = M_{xy} e^{i\omega t}. \] \hspace{1cm} (17)

where \( M_{xy}(t) = M_0 (1 - e^{-\gamma t/2}) e^{-\gamma t/2} \) is the magnitude of magnetization, and is exactly what MRI attempts to recover. This quantity captures magnetic properties of the material, such as proton density and relaxation times \( T_1 \) and \( T_2 \). Proton density and relaxation times \( T_1 \), \( T_2 \) are different for different tissues and are therefore the source of contrast in the images. An excellent discussion on designing pulse sequences, i.e., determining \( T_R \) and \( T_E \) based on the properties of imaged tissues, can be found in [10], pp. 485-493.

2.1.4 Conclusion

To summarise, magnetisation of a sample can be measured by tilting its vector away from the stationary magnetic field. If an rf pulse of Larmor frequency is applied to the sample, magnetic resonance occurs, and the magnetisation vector flips into the transverse plane. After the rf pulse is turned off, relaxation processes cause decay of the transverse magnetisation and recovery of the longitudinal magnetisation component to its original value.
The next section describes how the signal measured by the receiver coil can be used to reconstruct the 3D distribution of $\hat{M}_{xy}$.

### 2.2 Spatial encoding in MRI

If a receiver coil is placed along the object, the transverse magnetization induces current in the coil. This signal is the integral of the magnetization over the entire volume:

$$S(t) = \iiint M_{xy}(x,y,z,t) \, dx \, dy \, dz = \iiint \hat{M}_{xy}(x,y,z,t) e^{i(\omega(x,y,z,t) + \varphi(x,y,z,t))} \, dx \, dy \, dz \quad (18)$$

The task of spatial encoding is to vary frequency $\omega(x,y,z,t)$ and phase $\varphi(x,y,z,t)$ over the volume for every measurement in such a way that the original distribution of magnetization $\hat{M}_{xy}(x,y,z,t)$ can be recovered from a set of integrals [equation (18)]. This is achieved by applying a spatially variable (stationary in time) magnetic field $B'_{xy}(x,y,z) = B_{xy}(x,y,z)\hat{z}$ that induces spatial distribution of Larmor frequencies [equation (4)] over the volume.

Spatial derivatives of $B'_{xy}(x,y,z)$ determine local resolution of the image. Since constant gradients yield a uniform resolution over the volume, as well as optimal bandwidth characteristics of the current pattern in the gradient coils, linearly varying magnetic fields are commonly used in MRI. In the remainder of this document, we denote the spatially constant derivatives of $B'_{xy}(x,y,z)$ as $(G_x;G_y;G_z)$. Three different types of spatial encoding are commonly used in MRI: selective excitation, phase encoding and frequency encoding.
2.2 Spatial encoding in MRI

2.2.1 Selective excitation

If a spatially varying magnetic field $B'_0$ is present in conjunction with an rf pulse that contains a selected set of frequencies, magnetic resonance will occur only in a sub-volume whose Larmor frequencies are included in the rf pulse. This technique is commonly used for slice selection in MRI. In this case, $B'_0 = B_0 + G_z \hat{z}$, and the rf pulse contains a single frequency $\omega$. Only spins in a slice defined by

$$z = \frac{\omega}{\gamma G_z} \frac{B_0}{G_z}$$

(19)

are affected by the rf pulse. It can be shown that the profile of the slice can be approximated by a Fourier transform of the pulse function [11]. Thus to select a slice of uniform thickness, sinc rf pulses are used. The signal generated after the pulse is an integration over the selected slice (plane).

2.2.2 Phase encoding

After the excitation pulse, the distribution of transverse magnetization in the sample is essentially 2D. If we apply a linear field of gradient $G_y$, the Larmor frequency distribution will also be linear in $y$. This will cause variation in the phase of magnetization. After time $\tau$, the phase of point $(x, y)$ is determined by

$$\varphi(x, y) = (\omega(x, y) - \omega_y)\tau = \gamma G_y y \tau$$

(20)

After the gradient is switched off, the precession frequency returns to a constant value over the plane, while the phase remains proportional to $y$. 
2.2.3 Frequency encoding

If a constant gradient $G_x$ is applied to the sample, the frequency of precession will change linearly with location:

$$\omega(x, y) = \gamma G_x x$$

(21)

If the signal is read off while this gradient is on, contributions of voxels at different locations will have different frequencies.

Figure 2: Pulse sequence example for the standard Fourier encoding scheme.

If three encoding steps are performed in a sequence as shown in Fig. 2, at time $t$ after the beginning of the frequency encoding pulse $G_x$ the transverse magnetization of voxel $(x; y)$ in the excited plane is given by

$$M_{xy}(x, y, t) = \hat{M}_{xy}(x, y, t)e^{i(\omega(x, y)t + \phi(x, y))} = \hat{M}_{xy}(x, y, t)e^{i(\gamma G_x x + \gamma G_y y)}$$

(22)

Inducing a signal

$$S(t) = \iiint M_{xy}(x, y, t) dx dy = \iiint \hat{M}_{xy}(x, y, t)e^{i(\gamma G_x x + \gamma G_y y)} dx dy.$$  

(23)

If we denote

$$k_x = -\gamma G_x t, \quad k_y = -\gamma G_y t,$$

(24)

then

$$S(t) = S(k_x(t), k_y) = \iiint \hat{M}_{xy}(x, y, t)e^{-i(k_x x + k_y y)} dx dy = F_M(k_x, k_y)$$

(25)

where $F_M$ is the Fourier transform of the magnetization at time $t$. The phase and the frequency encoding steps essentially "tag" every location in the
excited slice with a distinct pair of a phase and a frequency of the magnetization precession. The integral of this pattern is the Fourier transform of the magnetization. The signal measured during one such iteration produces a row in the spatial frequency space \((k_x, k_y)\), also referred to as k-space in MRI literature. After repeating this process several times for different values of \(G_y\) and completing matrix \(S(k_x, k_y)\), the image of transverse magnetization can be recovered by applying the inverse discrete Fourier transform. This image is a single slice in the volumetric MRI scan of the sample. To obtain all the slices, the process described above has to be repeated for different values of the excitation frequency \(\omega\).
2.3 Source of image contrast

Magnetic resonance imaging offers a variety of sensitivities to physiological parameters of tissue, allowing tissues and pathologies to be delineated on the basis of differences in the local physico-chemical microenvironment. By appropriate choice of pulse sequence and parameters, it is possible to make image contrast dependent [16], for example, on intrinsic tissue spin relaxation times ($T_1$ and $T_2$), on local blood flow and perfusion, on water diffusion and on chemical and micro-structural properties of the tissue.

2.3.1 Relaxation time - T1 & T2

Relaxation can be considered in two parts: recovery toward equilibrium (longitudinal) alignment and transverse decay. Both of these processes can be described mathematically by exponential terms [see earlier section on “Relaxation” for mathematical analysis], governed by time constants, $T_1$ (for longitudinal recovery) and $T_2$ (for transverse decay). It is these parameters, $T_1$ and $T_2$ which are properties of the water microenvironment and thus of different tissue types, that lend MRI its inherent power to distinguish between different tissues (even of similar density). $T_1$ and $T_2$ differ between tissues because the physico-chemical microenvironments of tissues differ (especially water mobility and the presence of microstructures, macromolecules and membranes) [16].

Figure 3. Example of $T_1$-weighted and $T_2$-weighted images [23]. $T_1$-weighted image has good tissue contrast and the CNS fluid in the ventricles appears dark. Oppositely, the $T_2$-weighted image has less tissue contrast and the fluid appears bright.
2.3.2 Relaxation time - $T_2^*$

While describing $T_2$ (spin-spin) relaxation, it was assumed that we are able to apply a constant field $B_0$ across the sample. This is not in fact true. When a real magnet is built, there will always be some spread in $B_0$ values across the sample region [15].

There are two factors that contribute to the decay of transverse magnetization [13]:
1) molecular interactions (said to lead to a pure $T_2$ molecular effect)
2) variations in $B_0$ (said to lead to an inhomogeneous $T_2'$ effect)

The combination of these two factors is what actually results in the decay of transverse magnetization. The combined time constant is called $T_2$ star and is given the symbol $T_2^*$. The relationship between the $T_2$ from molecular processes and that from inhomogeneities in the magnetic field is as follows:

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{1}{T_2'}$$

The less homogeneous the field, the shorter is $T_2^*$ and the quicker the signal decays [15].
2.4 Pulse sequences

The soft tissue contrast provided by MRI has made it the most popular modality of choice for medical imaging. The unchallenged sensitivity of MRI for the evaluation of various diseases arises due to the dependence of MRI contrast and signal-to-noise ratio (SNR) on numerous intrinsic and extrinsic parameters. The most important intrinsic parameters, which depend upon individual tissue characteristics, include the spin-lattice relaxation time ($T_1$), spin-spin relaxation time ($T_2$), proton density, and velocity of moving protons. Extrinsic parameters affecting contrast and SNR are those chosen by the physician and/or technologist in performing the examination and include the echo time ($T_E$), repetition time ($T_R$), field-of-view, slice thickness, and resolution. In addition to the choice of these parameters, there is an almost bewildering choice of pulse sequences [24].

A pulse sequence defines the manner in which the radiofrequency pulses, which generate the detectable signals, and magnetic gradient fields, which provide the spatial encoding of the signals, are applied [24]. Fortunately, image contrast can be manipulated by both the choice of the extrinsic parameters and the selection of an appropriate pulse sequence to highlight regions of suspected pathology.

The purpose of this section is to understand the most common (basic and advanced) pulse sequences used in MRI.
2.4 Pulse sequences

2.4.1 Basic pulse sequence

2.4.1.1 Spin echo pulse sequence

A spin echo pulse sequence uses 90° radiofrequency pulses to excite the magnetization and 180° pulses to refocus the spins to generate signal echoes, see figure 4. It exists in many forms: the multi-echo pulse sequence using single or multi-slice acquisition, the RARE (rapid acquisition with relaxation enhancement) pulse sequence, echo planar imaging (EPI) pulse sequences and the GRASE (gradient and spin-echo) pulse sequence are all basically gradient-echo sequences.

![Spin echo pulse sequence diagram](image)

<table>
<thead>
<tr>
<th>Image Contrast</th>
<th>$T_1$ Value</th>
<th>$T_2$ Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD-weighted image</td>
<td>Short (20 ms)</td>
<td>Long (&gt;1600 ms)</td>
</tr>
<tr>
<td>$T_1$-weighted image</td>
<td>Short (10-20 ms)</td>
<td>Short (300-600 ms)</td>
</tr>
<tr>
<td>$T_2$-weighted image</td>
<td>Long (&gt; 60 ms)</td>
<td>Long (&gt; 1600 ms)</td>
</tr>
</tbody>
</table>

*Note:* with spin echo imaging no $T_2^*$ occurs.
2.4.1.2 Gradient-recalled echo pulse sequence

Of all the pulse sequences used in MRI, the collection of sequences that fall under the general classification of gradient-recalled echo (GRE) sequences is the largest and has the most confusing list of names [24]. This problem is only compounded by the fact that each manufacturer of clinical MR scanners has its own (lengthy) list of GRE sequences.

In general, GRE sequences are significantly faster than SE sequences for obtaining proton density-weighted images, $T_1$-weighted images, and images similar to $T_2$-weighted images. Without going into a lengthy discussion of the physics of these sequences [29], the GRE sequences differ from the SE sequences because of the absence of the refocusing radiofrequency pulse (180° pulse). Furthermore, the single radiofrequency pulse in the GRE sequences is typically less than the 90° pulse used in SE sequences. This allows for a reduction in scan time, but at the price of SNR and increased artefacts due to changes in magnetic susceptibility between tissues. For example, at the interface of bone and tissue or air and tissue, there is a prominent loss of signal that gets worse as $T_E$ is increased. These artifacts, most noticeable at the skull base, generally preclude the use of GRE sequences for obtaining "$T_2$-like" image contrast. Furthermore, the advent of FSE sequences for fast $T_2$ weighted image acquisition has made the use of gradient echo sequences for obtaining $T_2$-like images nearly obsolete.

The speed of GRE sequences has made them the technique of choice for obtaining $T_1$ weighted images from a large number of slices or from a volume of tissue as opposed to slice-by-slice. GRE sequences, therefore, are frequently used for $T_1$ weighted three-dimensional (3D) volume data acquisition that may require 128 slices each typically 1 – 2 mm thick.
CHAPTER - 3

3 Framework for MRI Database Tool

3.1 Why an MRI database tool?

From the previous two chapters it is clear that medical imaging data are multimodality in nature and can come from different acquisition systems or from image processing systems themselves. The data are usually two-dimensional images, three-dimensional images, or multidimensional sequences of these. The imaging data can also be in the form of curves or regions of interest. In order to be able to make these data available to end users, so that they may manipulate, store, analyse and manage these images within a simple, secure and easy-to-use graphical user interface (GUI), it is of fundamental importance to specify the range of data, its structure and interrelationships, as well as the parameters that hold vital information about the characteristics of the images.

Since a single study for a cancer patient might produce well over a thousand MR images, the quality of diagnosis and treatment of cancer does not only depend on the image manipulation and analysing tools but also on how effectively these images are tracked and managed. Hence, an integrated tool for managing MR images effortlessly, combined with a simple-to-use image storing, browsing and manipulation tool, is essential.

By further integration with existing (or legacy) medical electronic patient record (EPR) systems, MR image database tool will create a single flow of information, reduce duplication, improve collaborations among different user groups and increase overall team efficiency. This integration with the EPR system will also bring benefits to medical professionals such as searching MR Image database via patient name, national insurance number or any EPR
field; and giving access to image data to a wider range of physicians or clinicians or researchers via LAN, WAN, Intranet or Extranet.

3.2 Main objectives

The main objective of this research work is (1) to analyse requirements and develop a framework for a MR image manipulation tool within a relational database management system (RDBMS) environment, and (2) to research, design and develop a prototype system based on this framework and test its functionality with real-life data sets to identify its usefulness within the clinical environment.

In this chapter, the research and development of framework for MR image manipulation database tool and the necessary standard that is devised to achieve it, is discussed in great detail.

The framework, which includes the definition of a standard (generalised) database tool to acquire raw MR images from files, process and store both raw and processed MR images and related parameters in a powerful RDBMS, to provide GUI functionalities to search MR images by parameters, and to carry out various quantitative analysis on selected images with the help of plug-in analyser tools and store the findings back into the RDBMS for future comparison, will be specified in great detail.

In Chapter 4, the designed prototype of MR image manipulation database tool is discussed and the results are presented.
3.3 Functional Specification – development of a standard

The principle of development process of MR Image Manipulation Tools within a RDBMS environment can be described as to accept the raw MR images which are acquired from an MRI scanner, and put them through an image import routine to structure the image information in a standard format, and store the processed image information together with the necessary image parameters into a well defined database. Furthermore, it is necessary to provide GUI functions to search and manipulate stored images and parameters. This principle is illustrated in Figure 7.

Figure 7: principle of developing MR image manipulation database tools
The main areas where there are needs to develop standards, are:

(1) **Import routine** – to specify how to process raw images before storing into database as well as to specify which parameters such as image characteristics and patient information need to be imported, quantified and stored within RDBMS;

(2) **Data storage definition** – to develop guidelines to define various data types within RDBMS to hold physical\(^3\) information which can be searched, selected and retrieved in a constant speed without going through any lengthy conversion routine;

(3) **Search and retrieval** – to specify image search, selection and retrieval process which are useful and easy to use;

(4) **Image Manipulation tools** – to specify a set of rules that can be used by researchers from many disciplines to develop new image manipulation tools to assist their research and expand the capabilities of manipulation tools.

(5) **Graphical User Interface** – to specify a set of rules that can be used by programmers to develop new applications for image manipulation tools. Graphical user interface (GUI) will be the key human-machine interface (e.g. interfacing search tools via image parameters) and enable researchers to carry out their research in an environment where they feel more comfortable. Thus, it is important to document all requirements that will assist the researchers and reduce unnecessary repetitive works.

The above will be discussed in greater length in the next five sections.

---

\(^3\) Physical information means stored data as flat files within a computer hard disk as opposed to a managed environment such as RDBMS.
3.3 Functional Specification – development of a standard

3.3.1 Import routine

MR images, after acquiring from an MRI scanner, are often kept in a raw image format. Raw MR images are converted in run-time into suitable visible format with the help of various commercial MR image viewing software tools such as Interactive Data Language (IDL) for display and investigation. So the main task for import routine is to process the raw MR image information, quantify vital image characteristics and prepare images to be stored in RDBMS, which then can be displayed without further conversion work. This speeds up enormously the response time of the subsequent image selection procedure. The original data are retained in the database for subsequent analysis.

The import routine GUI should cover the following functionalities:

- Easy adjustment of import routine to work with data acquired from different MR Scanners.
- Easy adjustment of import routine to work with different RDBMS.
- Ability to select more than one raw image data (files) at a time, preferably from a directory where MR images are stored, for processing.
- Ability to pre-process images to be stored into the selected RDBMS.
- Ability to accept further patient information to link the image data with patient records.
- Ability to accept information for physician or clinician who will carry out image analysis. This is to introduce extra information which are not available within MRI data.
- Creation of a log of activities that import routine performs while processing raw images and view the log.

The principal functionalities of the import routine are illustrated in figure 8.
The above figure shows two functions for (1) separating image parameters and (2) separating entire image header. These are necessary because MR raw image data contains all image parameters within the image header. For a particular study or research it is not necessary to have access to all parameters. Separating all parameters from header information and storing them within the RDBMS will slow down the import routine and data processing. So only required parameters should be separated, indexed and stored in the RDBMS, and the entire header that contains all parameters should be kept in a BLOB. In this way, any parameters, which are not indexed but may be necessary in the future study, can be separated from the header information.
The design of an import routine may involve selecting more than one raw image file within a visual user graphics interface and passing them through an image processor that performs following actions automatically:

- Select raw image data file and read raw image data from the file sequentially.

- Process raw image data and prepare them for storage within RDBMS in a format that can be displayed by different operating systems (e.g. Windows, Linux, and Unix) without the need of further processing.

- Format image data to construct a binary large object (BLOB) that can be stored into database and retrieved from database independently using sequential query language (SQL).

- Store raw image in a database BLOB field retaining all its original image characteristics including the entire header information. In this way, if required, any future analysis can be carried out on the original image.

- Extract image header and image configuration parameters such as maximum intensity, scanner information, echo time and so on from the raw image file and process them to be stored within the RDBMS. The parameters are stored in such a way that they keep links to associated image data. Separating these parameters and storing them together with the image data will enable users to search and identify the images very quickly.

- Associate image data with patient record information.
The import routine also has the capability to store processed image data in an image database that can be defined by the user. That means, users at the beginning of the import process can specify where to store the processed image data or which image database to use for storing images. This flexibility will enable clinicians to collate or merge many image data sets into a single image database and use image manipulation tool to analyse them under one framework.

Before the processed image can be successfully stored in a relational database management system, the physical data device needs to be created with the appropriate data structures to hold image data as well as image associated parameters.

The design of database structure is discussed in the following section - the storage mechanism.

---

4 Physical implies the actual computer data storage such as disk, digital mass storage media and so on.

5 Data Device is a special binary data file that RDBMS uses to store database meta structure information, data type, data capacity, data size and so on.
3.3.2 Structure of MR image database storage

Once raw MR images are processed and prepared by the import routine for storage within a Relational Database Management System (RDBMS), the import routine needs to know exactly where and how to store them. This can be achieved by defining the data structure of physical storage mechanism.

Most RDBMS are capable of handling a large number of data, the physical disk space is the limit. Nevertheless, careful planning and optimisation of data structure can greatly increase performance while storing, searching and retrieving from RDBMS. Hence we need to lay down some basic requirements for how the database should be structured and how MR images and related information should be stored.

The MR image database storage should be structured to hold the following information:

- **Original MR image data** – the original MR image should be stored in a binary large object (BLOB) to retain all its original characteristics.
- **Pre-processed MR image data** – a version of the image, pre-processed to give optimum response time in other parts of the application should be saved as a separate BLOB.
- **Calculated MR images** – any calculated images should be stored within BLOB fields.
- **Image header information** – the entire image header information should be stored in a BLOB field.
- **Individual image parameters** – most frequently used image parameters should be separated from the header information and stored separately in a quantifiable data field (for example: numeric information should be stored in numeric data field and text information should be stored in character or string field and so on) and linked with relevant image. This will allow quick search facility based on parameters and locate relevant images very quickly.
- **Link to patient record system** – a patient identification record should be stored to link MR images with other electronic patient record information system.

- **Indexing** – MR image data must be indexed on parameters to provide speedy search facility.

- **Flexibility to expand** – MR image database should have the flexibility to expand with the number of imported image. This is important when selecting the right RDBMS as many commercial database systems can have limitation to grow database storage automatically.

A simplified data structure of storage mechanism is illustrated in figure 9.

**Figure 9:** A simplified data structure and entity relationship of RDBMS storage mechanism.
3.3 Functional Specification – development of a standard

**Image Data** is a table within RDBMS that holds all images and crucial information such as image parameters (e.g. echo time), patient identification number (PID), date, time and so on, which are directly related to the images. The structure of Image Data should be flexible enough to hold enough information, which are specific to certain clinical study or research; and to expand its capacity to hold more information by using SQL tools.

**Selected Image** is a table within RDBMS that holds information of all images, which are selected for a particular study or analysis. Selected Image table does not keep duplicated image data of the source image, but it maintains a direct reference (i.e. Image ID) to the Image Data table. It also maintains a group field identification number (i.e. Group ID), so that all images of a particular group can be queried and identified accurately. By adjusting database indexing constraints\(^6\) of the Selected Image table, it should be possible to include a particular source image from “Image Data” table allowing it to be included in more than one study but restricting it to be included more than once in this same study. In other words, the Selected Image table should have the capability to keep multiple references of the same source image that exists in Image Data table. This is also denoted by an “\(\_m\)” at the left top corner of the Selected Image table. A grey line linking Image Data table to Selected Image table indicates that Selected Image table keeps references to the Image Data table. The “\(\_i\)” at the left end of the grey line and “\(\_m\)” at the right end of the grey line express that an image from Image Data table may be referenced more than once by the Selected Image table.

**Image Group** is a table within RDBMS that holds information of all image groups. The Image Group table needs to exist because it provides a mechanism to name each study or analysis uniquely. It also holds all selected

---

\(^6\) Database Indexing Constraints define rules that a RDBMS actions automatically when information is written into, edited or deleted from the database. For instance, unique constraint can be added on “Image ID” and “Group ID” fields in Selected Image table in Fig. 9 to prevent same image to be included more than once in the same image group. This is how a well-designed RDBMS can improve the quality of overall study.
images within the group by keeping references to the Selected Image table. Another important role for the Image Group table is to keep track of the calculated image by keeping references to Resultant Image table (see next section).

**Resultant Image** is a table within RDBMS that holds information of all calculated images. Because the Resultant Image table is referenced by Image Group table, and Image Group table keeps references to Selected Image table which keeps reference to source images in Image Data table, the RDBMS is now able to respond to any sequential query language (SQL) properly and provide support to search engine, provide images to the graphical user interface (GUI) for display purpose, provide images to the plug-in modules to perform analysis and so on. This also facilitates the possibility to multi-layer nesting of image analysis, plus brings added benefits such as:

1. eradicating the need to copy and paste source/raw images manually to a separate directory before carrying out any image analysis, hence saving disk space, time and cost,
2. removing the need to store calculated images manually as the Resultant Image table accommodates the storage space for the calculated image, and
3. most importantly keeping links between source images, calculated images, patient records and crucial image parameters which eventually will enable clinicians and researchers to carry out more advanced analysis which was very difficult to carry out manually earlier.

**Data fields** - defining data fields of a particular table within RDBMS is as important as defining image data table. Image database can grow very fast, so by fine tuning data fields, one can achieve performance gains in terms of speed of storing images, querying image data, and performing certain analysis, plus one can save storage space and time. For instance - the individual data fields of Image Data table can be specified as shown in Table 1: Image Data Definition.
Table 1: Image Data Definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGEID</td>
<td>INTEGER</td>
<td>Defines table key field</td>
</tr>
<tr>
<td>FILENAME</td>
<td>VARCHAR(50)</td>
<td>Holds original image file name only</td>
</tr>
<tr>
<td>FILEFULLNAME</td>
<td>VARCHAR(80)</td>
<td>Holds original image file path and name</td>
</tr>
<tr>
<td>IMAGE</td>
<td>BLOB</td>
<td>Holds processed image data</td>
</tr>
<tr>
<td>IMAGE ORIG</td>
<td>BLOB</td>
<td>Holds original raw image data</td>
</tr>
<tr>
<td>FILEHEADER</td>
<td>BLOB</td>
<td>Holds entire image header information</td>
</tr>
<tr>
<td>MAXINTENSITY</td>
<td>SMALLINT</td>
<td>Holds image intensity</td>
</tr>
<tr>
<td>IMAGEWIDTH</td>
<td>SMALLINT</td>
<td>Holds image width</td>
</tr>
<tr>
<td>IMAGEHEIGHT</td>
<td>SMALLINT</td>
<td>Holds image width</td>
</tr>
<tr>
<td>IMAGEPIXELSIZE</td>
<td>SMALLINT</td>
<td>Holds image pixel size</td>
</tr>
<tr>
<td>ECHOTIME</td>
<td>SMALLINT</td>
<td>Holds echo time</td>
</tr>
<tr>
<td>FILESEQUENCE</td>
<td>VARCHAR(80)</td>
<td>Holds image file sequence information</td>
</tr>
</tbody>
</table>

Further explanation of specified fields are discussed here.

**ImageID** – This is specified as the table key field. It is important that every image can be identified uniquely within the image data table. It is specified to be an integer type, though it could have been character or alpha-numeric type, because it will be easy to generate an integer number while implementing the system; and an integer takes up less physical space than characters or alphanumeric data type hence the performance will be better.

**FileName, FileFullname and FileSequence** – Variable character data types are specified for these fields as the length of these are not known at design time. So a variable character data type will grow as the length of data will grow and eventually should save system resources such as disk space and processing. The "file name" is separated from the header information and stored in a separate field so that the users can search images by their "file name".

---

7 Key field means the value of this will be unique within all data rows of the image data table. Once a value is assigned to this field it should not be changed. So every image within the RDBMS can be uniquely identified by referring to this field.

8 Characters data type will take up 8 bits (i.e. 1 byte) for every digit. So for a 10 digits characters or alphanumeric field RDBMS has to accommodate 80 bits, where an integer data field only requires 64 bits.
Image, Image_org and Image FileHeader – are specified as binary large object (BLOB). A BLOB is most suitable for storing image data as they are in the form of binary information.

MaxIntensity, ImageWidth, ImageLength, ImageSize and EchoTime – are specified as Small integer\(^9\) because their value will never exceed more than 16 bits numeric value.

To improve overall performance of the MR Image Manipulation Tool within the RDBMS environment, it is important to identify all possible fields, which will be used frequently to query the system, and create appropriate indexes\(^{10}\). So it is important to define following indexes within MR Image database to improve its performance:

Indexes within Image Data table
- Image ID – to be unique for each image. This is also the primary key\(^{11}\) for Image Data table.
- Parameters – such as echo time, file sequence, image pixel, maximum intensity, file name and so on.
- Patient Record ID – to link with Electronic Patient Record System.

Indexes within Selected Image table
- Image ID
- Image Group ID
- Date of Selection

---

\(^9\) Small Integer is a 16 bits numeric field.

\(^{10}\) Indexing is a mechanism that RDBMS uses to sort and store information within the physical database storage space. The main objective behind indexing is that it allows the RDBMS to retrieve indexed information very fast. Once indexes are defined, RDBMS will apply them as information are stored and queried.

\(^{11}\) Primary key is a unique index constraint that is used by the RDBMS to identify and update a row of information within a database table. Once the primary key field is created, it remains unchanged throughout the lifecycle of its existence.
Indexes within *Image Group* table
- Image Group ID
- Image Group Name
- Patient Record ID
- Patient Name
- Patient Birth Date
- Date of Analysis Performed

Indexes within *Resultant Image* table
- Calculated Image ID
- Image Group ID
- Calculation Date
- Calculated Parameters
- Calculated Image Title
- Any quantitative data defined by the clinician that might be useful for future study or analysis.

The selected RDBMS should provide the facility to create any extra indexes that might be necessary at a later stage to fine tune the over-all performance of the MR Image manipulation tool. This should be possible without the need to recreate the entire database from scratch.

**Conclusion**
It is important to stress the importance of specifying the appropriate data storage to support the MR Image manipulation tool within RDBMS environment. Minor changes in specification of data structures; data field definition; constraints/rules; referential integrity and indexing can result in decreased or increased performance of the system. Thus before carrying out any changes, one should master the basic technique of database tuning. There are many books on the market on data structure, SQL and optimising
3.3.3 Search and retrieval

Fast search and retrieval of stored images is one of the key contributions to the entire MR image database tool development. Once raw images are processed and stored, the management of images and image related information are centralised and inter-linked within the RDBMS. So locating a single image or a group of images that satisfy certain criteria or parameters should be simple and fast. Looking at the advances of modern computer CPU (central processing unit) speed, a large number of images can be searched and displayed within a few milliseconds given the MR images are properly indexed and structured within RDBMS.

An MR image search and retrieval tool should satisfy following criteria:

- Facility to search images by any stored image parameters for example: echo time, file sequence, image pixel, maximum intensity, file name and so on.
- Facility to view searched images via the graphical user interface of MRI database tool.
- Facility to select images for further manipulation and store the references to selected images under a group name, which can be viewed at a later date without the need of searching again.
- Facility to view and extract image header information. It is also desired to have search facility for all image header information.
- Save search criteria for future use. This should eliminate the need to recreate the same search criteria more than once.
- Ability to use search facility on its own with multiple databases. A user-level interface for search engine should be able to hook into multiple databases that may be specified by the user, and carry out the search.
- Facility to incorporate search and retrieval functionalities within a dynamic linking library (DLL) format so that it can be used within any application.

### 3.3.4 Image manipulation tools (plug-ins)

The development of new MR image manipulation tool for quantitative image analysis will provide clinicians and researchers a new way to look at the MR image information. A range of image manipulation tools (plug-in modules) would allow properties such as $M_0$, $T_1$, $T_2$ and $T_2^*$ to be calculated from the previously selected group of relevant stored images from a particular database. Again, the calculated image or the findings can be stored back into the database, which can be compared with the similar entities at a later date.

To allow different clinicians to share and compare their findings with each other in the process of diagnosing illness or monitoring progress of a certain illness, the image manipulation tools need to be greatly generalised in the sense of how they select image data sets, how they manipulate selected data sets, what criteria are used and, how the findings are stored.

Two main areas need to be generalised while developing a plug-in module for data manipulation:

- **Input parameters** - a well-defined standard image selection procedure, which will return a number of parameters that can be used as the input parameters for the manipulation tool.

  For example: A $T_2^*$ plug-in module will require a set of image data. So the input parameter can be an SQL (sequential query language) string that defines a set of MR images that $T_2^*$ calculation requires to calculate $T_2^*$ image.
In addition, many algorithms require additional (non-image) parameters (e.g. default values), which could also be either stored in the database or passed as the input parameters.

- Output parameters – a well-defined standard image storing procedure, which will save calculated image(s) and parameters back into the database.

3.4 Technical Specification

The objective of technical specification is to define all aspects of technical requirements to achieve an MRI database tool.

The technical specification should define the followings:

- Specifying server and client operating system – it is possible that a particular implementation may have a Linux or Unix server and client side may use Windows XP Professional. Whatever the operating system is, the most important criteria are to maintain an overall compatibility. So one does not run into incompatibility between operating system and RDBMS.

- Specifying RDBMS – nowadays, there are many high performance, heavy duty relational database management systems in the market that run on almost any operating system, such as Oracle, MS SQL Server, Borland InterBase, PostgreSQL. If money is a big problem, it is worthwhile to consider going for the open source version of PostgreSQL or InterBase\(^{12}\) RDBMS. Open source database system

\(^{12}\) InterBase version 6.0 is the only version that was open source product. Any earlier and later versions are not open source, hence a license fee per user need to be paid before deploying these versions of InterBase.
can give users more control over the future development of the database tool as well.

The main disadvantage of most of the open source databases is that they do not support very large number of concurrent user access to the database. Typically when thousands of concurrent users need to have access to the same database server, open source database will become slow or fail to serve. Commercial databases such as MS SQL server and Oracle will retain good performance even under heavy demands from large number of concurrent users.

For a "research" environment, an open source database is thus likely to be sufficient, whereas a final deployment in the health sector might need a commercial product.

- Specifying database tool operating environment - while developing a plug-in tool it is possible to develop a cross platform application that can be accessed via the web. Again, if the web deployment is not required, a conventional application should be sufficient.

- Specifying development environment - a good object oriented programming language must be selected based on the desired deployment environment, RDBMS and operating system requirement. A programming language with the capability of generating 100% native machine code, and well-designed IDE\(^\text{13}\) (e.g. C++, Delphi) and the capability to design and deliver tool with very high standard, is highly recommended.

- Specifying a math library - while developing a plug-in module that involves processing complicated mathematical expression, it is

\(^{13}\) IDE – Interactive Development Environment (IDE) is a graphical user interface (GUI) for programming, designing and compiling applications.
important to select a math library that does not only provide good mathematical API\(^\text{14}\) but also fits in the programming environment. Selecting an appropriate math library can save much of the developing time.

### 3.5 Graphical user interface

Just as constructing the right MR image manipulation tool for one’s research is important, so is designing the right graphical user interface (GUI) that can support all functionalities to interact with the MR image manipulation tool. It is also important that the graphical user interface supports all functionalities of database connectivity as well as all functionalities of all MR image tools such as the search engine and import routine. For instance, an image import routine is designed to handle the import of images from both single image file and multiple image files in one go, but if the GUI for the import routine does not support selecting more than one image file, the capability of import routine will go unused and the user will waste time by selecting individual image files. This can lead to incompatibility and eventually the GUI will become more of a burden than an advantage to the researchers.

So while specifying the functionalities of a graphical user interface for any MR tool, it is important to follow these principles:

- *Functionality mapping:* the GUI must support all functionalities of the underlying MR image tool. It must maintain one-to-one relationship with the MR Image Tools.

- *Automation:* the GUI must automate all processes, which are possible either by selecting data from the database or from any other storage

---

\(^{14}\) API – Application Programming Interface (API) is a set of compiled routines in the form of program functions or procedures that can be interfaced and called upon to perform certain algorithm or calculations.
3.5 Graphical user interface

media. For instance: GUI for image search engine can populate all image parameters in a drop-down list, and to create search criteria, the user can select the desired parameter from the drop-down list rather than typing it in.

- **Accessibility:** the GUI must satisfy all user requirements such as right colour scheme, font-size, on-screen keyboards and so on. The GUI must be well designed so that both able and disabled users can use it efficiently.

- **Compatibility:** graphical user interface must be compatible with the users' operating system such as Windows XP Professional, Red Hat Linux, Mac or OS2.

- **Cross-platform operability:** More and more the same GUI for client application is becoming accessible from many different platforms such as applications designed under J2EE\(^\text{15}\) or Windows Browser based technologies. So if MR image manipulation tool needs to be accessed by users from different geographical areas, the GUI should be designed under technologies such as J2EE.

J2EE defines the standard for developing multi-tier cross-platform enterprise applications. The J2EE platform simplifies enterprise applications by basing them on standardised, modular components, by providing a complete set of services to those components, and by handling many details of application behaviour automatically, without complex programming. For more information on J2EE see reference [75]

3.6 Advantage and disadvantage

Here are the main advantages that an image database tool can deliver:

- The ability to process a large number of raw images in a very short time.
- The ability to store millions of images within single RDBMS. (RDBMS automated data management, unlike conventional file oriented image database where images are saved manually as separate image files).
- Fast access to stored MR image data via SQL.
- Excellent GUI functionality for image searching and image manipulation.
- The ability to develop very customised plug-ins to serve specific requirements. The true potential of MRI technology may be realised by future development of advanced plug-in modules.
- Creation of 3D model of a region of interest from image data sets stored in RDBMS.
- Better diagnostic tool means clinicians can reduce their time spent on managing and processing image information, and can maximise their time spent on diagnosis of the problem and treating the patients.

Here are some disadvantages:

- Difficulties to maintain the unified standard – the more experts and researchers are involved to develop many plug-ins and database tools, the more difficult it will be to keep hold of single standard.
- Skill set problem – because this is a very new concept, finding the right skill sets to develop plug-ins can be difficult.
- System compatibility – the fast advances of current information technologies can dominate the progress of steady development of MR image database tools. A plug-in tool that was developed on an older platform may not run on a newer version of the same platform. Thus creating more constrain on the progress of tool development, as well as on usability of the tool.
4 Demonstration of MRI Database Tool

4.1 Introduction

In this chapter the prototype of an MR Image database tool is presented that is developed in accordance with the framework described in chapter 3. It incorporates all the basic ideas concerning data import and storage, and demonstrates how it is possible to create plug-in module, written in variety of languages, and incorporate them in the RDBMS framework. This has been done to illustrate what can be achieved with the data from a typical multi-functional MR examination.

The prototype of MR Image database tool is designed based on the Borland Open Source RDBMS (i.e. InterBase version 6.x), and compiled with Borland Delphi\textsuperscript{16} 5 enterprise edition into an application that contains following system modules:

- A database selection module to specify where to store processed image data.

- An import module for image files generated by a Siemens Magnetom Vision MRI scanner. Image data files used from Royal Marsden Hospital.

\textsuperscript{16} Delphi is a programming language for designing application and its graphical user interface. Delphi is based on the typical Pascal programming language and object Pascal. It includes a very user-friendly interactive development environment for programming. It is also considered as a RAD (rapid application development) tool.
- An image search and selection module in dynamic link library\(^{17}\) (DLL) format as well as an integrated application module.

- A plug-in module that utilises the imported image and search tool to select images and calculate \(T_2^*\) and \(M_0\) images. The plug-in module stores the calculated \(T_2^*\) and \(M_0\) images back into the database.

The detailed source code for the above application and its functional modules is included with this thesis, see Appendix -A.

In this chapter, the process of developing the prototype of MRI Database Tool and its components will be discussed in detail, and graphical user interface (GUI) of the prototype will be illustrated. This chapter is mainly organised into four main sections:

1. **System specification** – documents the operating system, software and other technologies used for the project.
2. **Import routine** – documents the MRI dataset used, MRI database schema, MRI raw files selection and preparation, MRI import procedure, and associated GUI.
3. **Search and selection** – documents the development of search and selection of MR images within RDBMS.
4. **Plug-in module** – documents the calculation of \(T_2^*\) and \(M_0\) and the procedure for storing the calculated images into the RDBS and related GUI.

\(^{17}\) Dynamic link library (DLL) is a 100% compiled machine code that contains a set of functions or procedures that other applications can call during runtime. DLLs generally do not have a graphical user interface; instead, they are usually accessed by applications without any user intervention. When a function of the DLL is called by an application, the DLL then needs to be loaded in the memory. Once the DLL is loaded in the memory, it can be called by other applications without the need of reloading. This is what makes DLLs unique and different from conventional applications.
4.2 System specification

4.2.1 System used in designing MRI Database Tool

During the development process of MR image database tool, following software tools are used:

Table 2 - System specification

<table>
<thead>
<tr>
<th>Description</th>
<th>Used software or tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server operating system¹⁸</td>
<td>Windows 2000/2003 Server, SP2/SP1</td>
</tr>
<tr>
<td>Client operating system:</td>
<td>Windows 2000/XP Professional, SP2</td>
</tr>
<tr>
<td>RDBMS:</td>
<td>InterBase v6.x, open source version. InterBase v7.5, commercial version. InterBase® is the Borland Relational Database Management System (RDBMS).</td>
</tr>
<tr>
<td>Programming languages:</td>
<td>Delphi 5 for visual GUI: Import routine, Search and selection Plug-in for calculating ( T_2^* ) and ( M_0 ) image. C++ for compiling DLL for search and selection routine.</td>
</tr>
<tr>
<td>Network protocol:</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>Application capability:</td>
<td>LAN, WAN, VPN, Terminal Server Services.</td>
</tr>
<tr>
<td>MRI scanner:</td>
<td>Siemens Magnetom Vision, 1.5T</td>
</tr>
</tbody>
</table>

¹⁸ For a single user environment, server system can be the same machine as the client PC. But for multiple users environment a server computer loaded with database server is required.
4.2.2 RDBMS for MRI database tool

The main reasons why InterBase was selected to be the back-end database platform for this project are described as followings:

1. Cross-platform compatibility - InterBase® offers cross-platform compatibility between operating systems such as Windows®, Linux and Unix® platforms and hardware systems - without requiring recoding or support for multiple database back-ends.

2. Transaction support - Distributed transaction support and access to multiple databases is built into the architecture of InterBase.

3. Network connectivity - InterBase supports most popular network protocols such as TCP/IP® and NetBEUI®. In building the MRI database tool only TCP/IP is used to communicate between the graphical user interface at the client-end and the image database server.

4. InterBase database connectivity - InterBase provides direct connectivity API to perform operations such as SQL query, insert, select and delete on database. It also supports most popular database connectivity such as ODBC and JDBC.

**ODBC** stands for open database connectivity. ODBC provides a standardised set of rules for getting information to and from a database. An ODBC driver is a software interface that accepts standard SQL queries from applications and processes them to conform database specific rules.

---

18 TCP/IP stands for transmission control protocol and Internet protocol. This is the main foundation protocol for most of the network communications that are available today.

20 NetBEUI stands for NetBIOS enhanced user interface. In turn, NetBIOS stands for network basic input/output system. This is a network protocol mostly used by Microsoft system such as DOS and Windows, and can communicate only within local area network.
and then passes them to the database for performing these SQL tasks. ODBC is also responsible to accept the result data sets from the database and then pass them to the application. ODBC works as a mediator and eliminates the need for a programmer to learn the complex mechanism of connecting a database before he can interact with the database.

**JDBC** stands for Java database connectivity. This is very useful for creating an interface that needs to communicate via the Internet protocol and needs to work on multiple system platforms. JDBC is mainly an API, which consists of a set of Java classes, interfaces and exceptions and a specification to which both JDBC driver vendors and JDBC developers adhere when developing applications.

(5) **InterBase development** - InterBase integrates very closely with many object oriented programming languages such as C++, Visual Basic, Fortran, Borland® Kylix™ and Delphi™ (Object Pascal). InterBase also supports stored procedure, which is a very powerful embedded programming language within the RDBMS, and utilises the data processing power at the server-end. Stored processor can be utilised to create thin-client MRI data manipulation plug-in which can run via the web.

(6) **InterBase design tool** - InterBase provides a GUI design tool that can be used to design, maintain and expand the database of MRI tool.

(7) **InterBase scalability** - InterBase is capable of running from a standalone desktop PC to a complex multi-tier client/server environment. It can work with multiple database files. InterBase is also a multi-threaded database. This means that the database should perform better than other non-multi-threaded database when more concurrent users are active.
InterBase limits
Like all RDBMS InterBase has limits which need to be observed while designing and expanding database schema of the MR Image Database Tool.

Table 3 defines the limits of a number of InterBase characteristics. The values are design limits defined by Borland Inc, and in most cases are further restricted by finite resource restrictions in the operating system or computer hardware.

Table 3 – InterBase limits

<table>
<thead>
<tr>
<th>Characteristic/Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of clients connected to one server</td>
<td>InterBase performs very well under high transactions and low volume data environment and is able to serve up to 1000 users. The MR Image Manipulation tool is considered to be low in transactions and high in data volume. So it is highly recommended that the number of concurrent users to this system be kept less than 100. Should the number of concurrent users be more than 100, it is highly recommend to use other enterprise RDBMS such as MS SQL Server or Oracle.</td>
</tr>
<tr>
<td>Value:</td>
<td></td>
</tr>
<tr>
<td>There is no single number for the maximum number of clients the InterBase server can serve - it depends on a combination of factors including capability of the operating system, limitations of the hardware, and the demands that each client puts on the server</td>
<td></td>
</tr>
<tr>
<td>Maximum database size</td>
<td>It has been reported by the support team of UltraSoft Technologies Ltd that the InterBase database shows signs of getting slow once its size exceeds 4GB. If the database size exceeds 4GB, it is recommended that the database is divided into multiple data device files to maintain a size of less than 4GB for each database. InterBase provides functionality to attach multiple databases and treat them as one.</td>
</tr>
<tr>
<td>Value:</td>
<td></td>
</tr>
<tr>
<td>No limit is imposed by InterBase; maximum size is defined by the operating system.</td>
<td></td>
</tr>
<tr>
<td>Characteristic/Value</td>
<td>Note</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Maximum number of cache pages per database</td>
<td>While deploying MR Image Tool, client needs to be configured properly by allocating a more practical number of cache pages. Allocating a large number of cache pages will dramatically improve the speed of the application but may cause other systems to run slow. So the total size of cache pages should never exceed 50% of the total memory.</td>
</tr>
<tr>
<td>Value: 65,536</td>
<td>The only time multiple databases will be necessary, will be when the single database size will exceed 4GB.</td>
</tr>
<tr>
<td>Maximum number of files per database</td>
<td>The only time multiple databases will be necessary, will be when the single database size will exceed 4GB.</td>
</tr>
<tr>
<td>Value: By design, $2^{16} (65,536)$, because the files are enumerated with an unsigned 16-bit integer.</td>
<td>Because image database will be large in quantity, the limitation of hard disk will dictate the number of databases. For instance: a 80GB hard disk will be able to accommodate only 20 of 4GB database files.</td>
</tr>
<tr>
<td>Maximum row size</td>
<td>Image data are stored in a BLOB, which contributes only 8 bytes to this limit and remaining data are saved under different system table automatically. This makes InterBase unique as a database for this project.</td>
</tr>
<tr>
<td>Value: 64KB. Each BLOB and array contributes eight bytes to this limit in the form of their BLOB handle.</td>
<td>Because InterBase limits the maximum number of rows to $2^{32}$, the maximum number of images can be stored in a table is 4,294,967,296.</td>
</tr>
<tr>
<td>Maximum number of rows per table</td>
<td>A typical MR study may produce on average up to 1000 images (in our experimental data it is 777 images). So an image data table will be able to hold images for 4.29 million studies before it runs out of its capacity.</td>
</tr>
<tr>
<td>Value: By design, $2^{32}$ rows, because rows are enumerated with a 32-bit unsigned integer per table.</td>
<td>In the unlikely event that requirements exceed this limit, the database table needs to divided into multiple tables or multiple databases need to be catered for.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3 – InterBase limits (continues)

<table>
<thead>
<tr>
<th>Characteristic/Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum stored procedure size</td>
<td>Stored procedure is suitable for a small program that needs to manipulate a large number of data from a database. Because stored procedure reduces the amount of data communication data sent back and forth between database server and client application, it improves application performance. For instance: in this project a stored procedure “GenNewKey” (see Appendix B – Database Schema) is programmed to generate unique ID for Image data and other tables. As long as a table name is passed to this stored procedure, the next unique ID will be returned to the application.</td>
</tr>
<tr>
<td>Value: 48KB</td>
<td></td>
</tr>
</tbody>
</table>

What is a stored procedure?
A stored procedure is a function that is defined within the RDBMS. Stored procedure can be executed directly within the RDBMS. An application can pass a number of parameters (the input values) to a stored procedure, and basing the parameters, stored procedure can access relevant data from the database and process them, and finally return the result sets to the application.

Maximum BLOB size

<table>
<thead>
<tr>
<th>Page size</th>
<th>Max BLOB size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1KB</td>
<td>64MB</td>
</tr>
<tr>
<td>2KB</td>
<td>512MB</td>
</tr>
<tr>
<td>4KB</td>
<td>4GB</td>
</tr>
<tr>
<td>8KB</td>
<td>32GB</td>
</tr>
</tbody>
</table>

Please note: A BLOB is a stream of many segments. The maximum BLOB segment size is 64KB

The test images used in this project have a size of 256KB (256x256x4). So the minimum database page size (i.e. 1KB) would be sufficient.

Because the database page size has great effect on how fast large amount of data can be saved and retrieved from the database, the maximum page size 8KB is used in this project to achieve the best speed possible for insertion, search and retrieval of image data.

Maximum number of indexes per table

<table>
<thead>
<tr>
<th>Value: By design, $2^{15}$ (65,536)</th>
</tr>
</thead>
</table>

Indexing is a key factor to increase performance of image search and retrieval.

So it is highly recommended to use as many indexes as possible while expanding the capacity of MR image manipulation tool.
Besides bearing in mind the above database limits, the performance of the MR image database tool greatly depends on the technique of how the system is designed, complied, deployed and operated.

To identify the best speed and performance of the system, a number of setups and configurations has been tried out and the following minimum configuration has been identified:

- **Hardware configuration**
  
  For single user deployment the RDBMS (image database) server and the GUI are configured to run on a single PC with minimum of 256MB RAM\(^{21}\), 10GB hard Disk, 1GHz CPU\(^{22}\) speed, 1024x768 pixels screen resolution and 32bits colour configuration.

  For multi-users deployment the RDBMS server has been loaded on a MS Windows Server with minimum of 128MB RAM and 12GB SCSI\(^{23}\) Hard Disk, and GUI application was loaded on 2 PCs.

  For single user deployment: It has been observed that PC with more memory rather than CPU speed shows better performance when searching and viewing images from the database, but calculating \(T_2^*\) images becomes faster in both cases when memory and CPU speed are increased.

  For Multi-users deployment: It has been observed that increasing server memory rather than increasing server CPU speed, has greater impact on

---

\(^{21}\) RAM — random access memory.

\(^{22}\) CPU — stands for central processing unit.

\(^{23}\) SCSI — stands for small computer system interface. SCSI hard disks are more resilient than IDE (integrated drive electronics) hard disks.
the function for search and retrieval of images. Increasing memory on client PC has faster response time for viewing images.

- **Operating system configuration**
  The image database of MR image manipulation tool primarily designed with InterBase RDBMS V6.0 (open source version) and tested under Windows 2000 Professional workstation and Windows NT4 Server and passed all tests (such as database connectivity, search and retrieval speed and so on) successfully.

  When tested under window XP professional and Windows 2003 .NET server, the database connectivity had been extremely slow. This was due to the recent changes in TCP/IP socket made by Microsoft. The slow connection problem has been resolved by Borland in next version, i.e. InterBase version 6.5.

- **Network configuration**
  Network configuration is only relevant for multi-users deployment. The communication between GUI front-end and image database happens only via TCP/IP protocol. It has been tested under 100MBPS (mege bits per second) Ethernet connection within LAN (local area network) and the performance with minimum hardware is good.

- **Application design techniques**
  The GUI application for this project is designed with Borland Delphi 7 to generate 100% compiled executable machine codes. Though it is a windows 32 bit application and like all windows applications, it has to depend on Microsoft windows core kernel functionalities, it has no dependencies on any other programs such as ActiveX or DLL. This makes the GUI very easy to be deployed and maintained.
4.2.3 Used MR image datasets

In this research project two sets of prostate MR image data from Royal Marsden Hosp, Sutton are used and have the following details:

Table 4 – MR image dataset details

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of images in a set</td>
<td>777 images</td>
</tr>
<tr>
<td>MRI scanner used</td>
<td>Siemens 1.5T Magnetom Vision</td>
</tr>
<tr>
<td>Typical image pixel size</td>
<td>256x256 pixels</td>
</tr>
<tr>
<td>Bytes of data per pixel</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Typical image size</td>
<td>128 KB (=131072 bytes)</td>
</tr>
<tr>
<td>Image header size</td>
<td>6 KB  (= 6144 bytes)</td>
</tr>
<tr>
<td>Single image file size (inc. header)</td>
<td>134 KB  (=137216 bytes)</td>
</tr>
</tbody>
</table>
4.3 Import routine

Before MR images can be successfully imported into RDBMS, a physical data device (PDV) needs to be created. PDV must map the data storage requirement of the imported images and their associated parameters, as well as any user requirements such as search criteria, patient id and so on. Once the database is ready, the MR images need to be extracted from their raw format and processed and prepared to match the database criteria for storing.

To give a thorough understanding of how the import routine is designed and implemented, the main functional modules will be discussed in the following sections. A GUI related to the import routine will also be demonstrated throughout these sections.

4.3.1 MR image database creation

Once the database structure is defined, the MR image database can be created with the help of RDBMS tool provided by the vendor. In this project, IBConsol.exe, a GUI tool provided by Borland, is used to create the database.

Like all RDBMS, InterBase uses generic SQL to create all database related entities such as databases, tables, fields, indexes, primary keys and referential integrity.

Creating the blank database

By executing SQL code shown in Schema 1, with the help of RDBMS tools (i.e. IBConsol.exe) the blank database device for the MRI tool will be created. The blank database does not contain any storage definition for holding images or any other information such as tables, fields and indexes.

\footnote{Map implies here that the prepared image information and database storage structure must maintain one-to-one relationship. Otherwise, the import routing will fail to function.}
4.3 Import routine

Schema 1 – SQL code for creating physical database device.

SET SQL DIALECT 3;
CREATE DATABASE
  'mri_srv:e:\dev\hb\unis\imageproc\data\IMAGEDB.GDB'
  PAGE SIZE 8192
  DEFAULT CHARACTER SET
COMMİT WORK;

While creating the database SQL DIALECT is used to take advantage of the latest SQL definitions. SQL DIALECT 1 is used only for backward compatibility.

'mri_srv:e:\dev\hb\unis\imageproc\data\IMAGEDB.GDB' is specific to InterBase. It means that the database will be created in "e:\dev\hb\unis\imageproc\data" directory on "mri_srv" image server and the name of the database device file will be "IMAGEDB.GDB".

DEFAULT CHARACTER SET - A default character set determines: (1) what characters can be used in CHAR, VARCHAR, and BLOB text fields and (2) the default collation order that is used in sorting a table column. The collation order determines the order in which values are sorted.

While creating the database device, a default character set was not specified. This means that there is no character set assumption for the columns; data is stored and retrieved just as it was originally entered. If this database was to be used with European languages, this default

25 SQL DIALECT determines how various objects such as data types; tables; fields; indexes; including their structures, behaviour and naming conventions, are defined within RDBMS. There are only two SQL DIALECT (i.e. 1 and 3) available in InterBase.
character set should have been "ISO8859_1" and command should have been DEFAULT CHARACTER SET ISO8859_1.

Creating table to store Images within RDBMS
Once the physical database is created, the necessary tables for storing MR images and related information can be defined and implemented.

Schema 2 shows the SQL code necessary to create the table that will hold the imported MR images and related information such as processed images that can be viewed without any further manipulation; original image data that retains all original characteristics; image parameters; study information and reference to patient record.

Schema 2 - SQL code for defining and creating table to store MR images.

```
CREATE TABLE "IMAGEFILES"  
(  
  "FILEID" INTEGER NOT NULL,  
  "FILENAME" VARCHAR(50),  
  "FILEFULLNAME" VARCHAR(80),  
  "EPR_ID" INTEGER,  
  "IMAGE" BLOB SUB_TYPE 0 SEGMENT SIZE 80,  
  "IMAGE_ORIG" BLOB SUB_TYPE 0 SEGMENT SIZE 80,  
  "FILEHEADER" BLOB SUB_TYPE TEXT SEGMENT SIZE 80,  
  "MAXINTENSITY" SMALLINT,  
  "IMAGewidth" SMALLINT,  
  "IMAGERIGHT" SMALLINT,  
  "IMAGEHEIGHT" SMALLINT,  
  "IMAGEPIXELSIZE" SMALLINT,  
  "ECHOTIME" SMALLINT,  
  "FILESEQUENCE" VARCHAR(80),  
  CONSTRAINT "IMAGEFILES_PK" PRIMARY KEY ("FILEID")  
) ;  
CREATE UNIQUE INDEX "IMAGEFILES_FN" ON "IMAGEFILES"("FILENAME") ;  
CREATE INDEX "IMAGEFILES_FS" ON "IMAGEFILES"("FILESEQUENCE") ;  
CREATE INDEX "IMAGEFILES_MI" ON "IMAGEFILES"("MAXINTENSITY") ;  
CREATE INDEX "IMAGEFILES_PI" ON "IMAGEFILES"("EPR_ID") ;
```

FILEID – is specified as the key field for IMAGEFILE. This is populated automatically with the unique value generated by the stored procedure
4.3 Import routine

"GenNewKey". A generic function has been written to pass the table name to retrieve its next unique key value.

**IMAGE** – This BLOB field is defined to hold processed MR image. The decision to store the processed MR image within RDBMS is one of the most innovative ideas in this project. Storing processed images in a BLOB field radically increases the speed with which images can be displayed. Because it is a pre-processed MR image, scaled in grey level for display to its original size and stored in the database, it can be retrieved like any other field and displayed on a screen without any further manipulation.

**IMAGE ORIG** – this BLOB is defined to store original raw MR image and retain all its characteristics. So that the original features can be analysed in the future if needed.

**FILENAME** – A unique index is created on the "FILENAME" field to make sure that the same image file is imported only once. In case a user attempts to import the same image file more than once, RDBMS will not store the duplicate image and generate an error to notify the user.

A number of non unique indices are defined on fields such as FILESEQUENCE and MAXINTENSITY to speed up the search and retrieval process for stored images. Please note that the primary keys are automatically and uniquely indexed by the RDBMS.

In this project only a minimum number of fields are created for testing and demonstration purpose, but any number of fields can be created or added to meet the future need of a specific study or research, even after the database has been populated with MR images.
Once the MR image database schema is defined and the physical database is created with the help of RDBMS tool, the database can then be connected via the user interface as shown in Figure-11 (callouts 1 and 2) and interact with it.

The database schema SQL code (also known as database metadata) can be exported at any point of time and can be used to create new databases. If it is necessary, images for a specific study can be grouped together and stored into a separate database. Because of that, the GUI for MRI database tool provides the facility to work with multiple databases, see figure 11, "database detail" area.

Before the MRI GUI tool can interact with the database, the GUI tool has to make a connection with the physical data device. A typical connection requires information about database server, data file path, database device file name, user ID and password. To save time, the GUI provides functionalities to save these parameters into a configuration file, which can simply be called upon at a later time, to make the connection with the database before interacting with it.
4.3 Import routine

4.3.2 MR Image files selection

One of the experimental raw MR image datasets used in this project contains 777 individual raw image files from a single scan. Before these images can be passed to the import routine for processing and storing into the database, there is a great need to identify a simple way to select them.

The MRI GUI tool used Microsoft Windows kernel API to explore and select multiple image files in one go. The relevant source codes are included in Appendix-A and the GUI is demonstrated in Figure 10.

![Figure 10 - MR image files selection process.](image)

To select multiple consecutive files hold down the "Shift Key", and click on the first image and then on the last image. To select random files, hold down the "Control Key" and click on the file.
4.3.3 MR Image preparation and import

In this project, an MR image import routine and associated GUI were designed to implement the following functionalities as specified in Framework in section 3.3.1 of chapter 3:

1. **Connecting to database** – routine to select an MR image database and establish connection to the database for writing image data;
2. **Reading MR data** – routine to select MR image files from disk or network drive and read image data from files;
3. **Storing original image** – routing to extract original raw image data and store it in the BLOB database field;
4. **Creating pre-processed image** – routing to create pre-process image from raw MR image data and store it in the BLOB database field;
5. **Extracting image header** – routine to extract header information from image data and storing it into the BLOB database field;
6. **Separating image parameters** – routine to separate the most important image parameters from the image header information and store them into the appropriate database fields;

The import routine is very specific to the raw MR image files. It needs to know the exact structure of the raw image file in order to read and process it. Again, the MRI scanner dictates the raw MR image file structure. So a thorough understanding of the image file structure is necessary.

To test all functionalities two sets of the MR images which are obtained by Siemens MRI Scanner are used. For detail on image dataset see Table 4.

The detail source code and functional routines for import routine are included in Appendix -A, *Import Routine*.
The graphical user interface that implements the import routine is demonstrated in Figure 11.

Following functionalities are implemented within the import routine:

1. Database selection for import routine.
2. Making sure database is functional.
3. Loading/selecting a series of raw image files from a directory.
4. Process and import images into the selected database.
5. Detailed import log that can be investigated later.

Figure 11 – GUI for implementing import routine
Table-5 shows a typical list of quantifiable image parameters that are extracted from the image header and stored in MR image database. But only a handful of these parameters (see SQL Schema 2) are indexed within the RDBMS for fast query of images. Using RDBMS tool, clinicians can specify more parameters to be indexed to suit their research work.

**Table 5 – Sub set of a typical list of imported image parameters (sub set)**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study date:</td>
<td>06/01/2003</td>
</tr>
<tr>
<td>Acquis. date:</td>
<td>06/01/2003</td>
</tr>
<tr>
<td>Image date:</td>
<td>06/01/2003</td>
</tr>
<tr>
<td>Study time:</td>
<td>09:44:07:180</td>
</tr>
<tr>
<td>Acquis. time:</td>
<td>09:44:09:330</td>
</tr>
<tr>
<td>Image time:</td>
<td>09:44:26:00</td>
</tr>
<tr>
<td>Data Set Subtype(1):</td>
<td>1</td>
</tr>
<tr>
<td>Data Set Subtype(2):</td>
<td>1</td>
</tr>
<tr>
<td>Modality:</td>
<td>2</td>
</tr>
<tr>
<td>Scanner Manufacturer:</td>
<td>SIEMENS</td>
</tr>
<tr>
<td>Hospital:</td>
<td>Royal Marsden Hosp, Sutton</td>
</tr>
<tr>
<td>Referring Physician:</td>
<td>AH</td>
</tr>
<tr>
<td>Station Ident.:</td>
<td>20831</td>
</tr>
<tr>
<td>Manufacturer Model:</td>
<td>MAGNETOM VISION</td>
</tr>
<tr>
<td>Contrast:</td>
<td>1</td>
</tr>
<tr>
<td>Scanning Sequence:</td>
<td>559126288</td>
</tr>
<tr>
<td>Slice Thickness (mm):</td>
<td>10</td>
</tr>
<tr>
<td>Repetition Time (ms):</td>
<td>15</td>
</tr>
<tr>
<td>Echo Time (ms):</td>
<td>6</td>
</tr>
<tr>
<td>Inversion Time (ms):</td>
<td>0</td>
</tr>
<tr>
<td>Number of Averages:</td>
<td>1</td>
</tr>
<tr>
<td>Imaging Freq. (MHz):</td>
<td>63.622784</td>
</tr>
<tr>
<td>Echo Number:</td>
<td>1</td>
</tr>
<tr>
<td>Nominal Fourier Lines:</td>
<td>128</td>
</tr>
<tr>
<td>Current Fourier Lines:</td>
<td>128</td>
</tr>
<tr>
<td>After 0 Fourier Lines:</td>
<td>83</td>
</tr>
<tr>
<td>First Fourier Lines:</td>
<td>-84</td>
</tr>
<tr>
<td>Acquisition Columns:</td>
<td>512</td>
</tr>
<tr>
<td>Reconstruction Coils:</td>
<td>512</td>
</tr>
<tr>
<td>Number of Averages:</td>
<td>1</td>
</tr>
<tr>
<td>Flip Angle (deg.):</td>
<td>30</td>
</tr>
<tr>
<td>Number of Prescans:</td>
<td>67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study :</td>
<td>1</td>
</tr>
<tr>
<td>Gap :</td>
<td>19222</td>
</tr>
<tr>
<td>Acquisition:</td>
<td>1</td>
</tr>
<tr>
<td>Image :</td>
<td>2</td>
</tr>
<tr>
<td>Sequence File Name:</td>
<td>/usr/appl/sequence/scout.wkc</td>
</tr>
<tr>
<td>Total Measr. Time (s):</td>
<td>34.39911348</td>
</tr>
<tr>
<td>Mag. Field Stren. (T):</td>
<td>1.4938025424281</td>
</tr>
<tr>
<td>Effective Rep. Time (ms):</td>
<td>15</td>
</tr>
<tr>
<td>Tol. Echoes:</td>
<td>1</td>
</tr>
<tr>
<td>Order of Slices:</td>
<td>1</td>
</tr>
<tr>
<td>Slice Thickness (mm):</td>
<td>10</td>
</tr>
<tr>
<td>Image Dimensions:</td>
<td>2</td>
</tr>
<tr>
<td>Image Rows:</td>
<td>256</td>
</tr>
<tr>
<td>Image Columns:</td>
<td>256</td>
</tr>
<tr>
<td>Pixel Size (Rows) (mm):</td>
<td>1.7578125</td>
</tr>
<tr>
<td>Pixel Size (Cols) (mm):</td>
<td>1.7578125</td>
</tr>
<tr>
<td>Patient Birth Date:</td>
<td>26-NOV-1936</td>
</tr>
<tr>
<td>Dataset Ident.:</td>
<td>STU/IMA 1/2</td>
</tr>
<tr>
<td>Image Rot. Ang. (rd):</td>
<td>0</td>
</tr>
<tr>
<td>Tol. Transmtr. Ampl.:</td>
<td>1</td>
</tr>
<tr>
<td>Transmitr, Calibr. (V):</td>
<td>379.779602050781</td>
</tr>
<tr>
<td>Recov. Total Gain (dB):</td>
<td>-0.139999900996144</td>
</tr>
<tr>
<td>Recov. Ampl. Gain (dB):</td>
<td>54.6660003862164</td>
</tr>
<tr>
<td>Recov. PreA. Gain (dB):</td>
<td>50.7790957926192</td>
</tr>
<tr>
<td>Recov. Calibr. Adj. (dB):</td>
<td>1.2089999592319286</td>
</tr>
<tr>
<td>Recons. Scale Factor:</td>
<td>0.014673850338244</td>
</tr>
<tr>
<td>Phase Grad. (mT/m):</td>
<td>2.69999996909863</td>
</tr>
<tr>
<td>Reeset Grad. (mT/m):</td>
<td>2.69999996909863</td>
</tr>
<tr>
<td>Select Grad. (mT/m):</td>
<td>2.400000000000000</td>
</tr>
<tr>
<td>Grad. Delay Time X:</td>
<td>42</td>
</tr>
<tr>
<td>Grad. Delay Time Y:</td>
<td>45</td>
</tr>
<tr>
<td>Grad. Delay Time Z:</td>
<td>41</td>
</tr>
</tbody>
</table>
### Table 5 – (Continues) Sub set of a typical list of imported image parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF Power Err. Indic.:</td>
<td>1.11774559153763</td>
</tr>
<tr>
<td>Spec. Absorb. Rate Lim.:</td>
<td>1</td>
</tr>
<tr>
<td>Absorb. Rate (WWKg):</td>
<td>0.06574289232023208</td>
</tr>
<tr>
<td>Absorb. Rate Deflected:</td>
<td>0</td>
</tr>
<tr>
<td>Energy Dose Limit:</td>
<td>100</td>
</tr>
<tr>
<td>Energy Dose [mJ/kg]:</td>
<td>0.01604611476569</td>
</tr>
<tr>
<td>Ener. Dose Detected:</td>
<td>0.0136262377737522</td>
</tr>
<tr>
<td>Calculation Submode:</td>
<td>1</td>
</tr>
<tr>
<td>Field of View Ratio:</td>
<td>1</td>
</tr>
<tr>
<td>Base Raw Matrix Size:</td>
<td>256</td>
</tr>
<tr>
<td>2D Phase Omp. Lines:</td>
<td>0</td>
</tr>
<tr>
<td>3D Phase Omp. Part:</td>
<td>0</td>
</tr>
<tr>
<td>Echo Line Position:</td>
<td>129</td>
</tr>
<tr>
<td>Echo Column Position:</td>
<td>188</td>
</tr>
<tr>
<td>Lines per Segment:</td>
<td>1</td>
</tr>
<tr>
<td>Phase Coding Direction:</td>
<td>1</td>
</tr>
<tr>
<td>Parameter File Name:</td>
<td>/usr/appl/proto/016/cli/cp-body-array/scout.prg</td>
</tr>
<tr>
<td>Image Position, x (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Image Position, y (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Image Position, z (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Im. Orient. R Axis, x:</td>
<td>1</td>
</tr>
<tr>
<td>Im. Orient. R Axis, y:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Orient. R Axis, z:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Orient. C Axis, x:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Orient. C Axis, y:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Orient. C Axis, z:</td>
<td>0</td>
</tr>
<tr>
<td>Location (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Field View Height (mm):</td>
<td>1.73635805196575E-310</td>
</tr>
<tr>
<td>Field View Width (mm):</td>
<td>450</td>
</tr>
<tr>
<td>View Direction:</td>
<td>3</td>
</tr>
<tr>
<td>ADC Offset Real. Part:</td>
<td>22.47255665</td>
</tr>
<tr>
<td>ADC Offset Imag. Part:</td>
<td>3.7955125</td>
</tr>
<tr>
<td>Transmit. Angle (deg):</td>
<td>25</td>
</tr>
<tr>
<td>Transmit. Ref. (V):</td>
<td>203.366843876706</td>
</tr>
<tr>
<td>Transmit. Ref. Gain (V):</td>
<td>106.437575478074</td>
</tr>
<tr>
<td>Rep. Freq. (Hz):</td>
<td>50000</td>
</tr>
<tr>
<td>Ref. Scaln Factor:</td>
<td>0.199678757003212</td>
</tr>
<tr>
<td>Total. Delay (us):</td>
<td>220</td>
</tr>
<tr>
<td>RF Watchdog Mask:</td>
<td>1</td>
</tr>
<tr>
<td>Parameter name</td>
<td>Value</td>
</tr>
<tr>
<td>Real Direction:</td>
<td>2</td>
</tr>
<tr>
<td>Image Pos. Seg. (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Image Pos. Cor. (mm):</td>
<td>-18</td>
</tr>
<tr>
<td>Image Pos. Tra. (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Im. Nor. Cos Dr. Sag.:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Nor. Cos Dr. Cor.:</td>
<td>1</td>
</tr>
<tr>
<td>Im. Nor. Cos Dr. Tra.:</td>
<td>0</td>
</tr>
<tr>
<td>Image Distance (mm):</td>
<td>0</td>
</tr>
<tr>
<td>Im. Pos. History Mask:</td>
<td>3</td>
</tr>
<tr>
<td>Im. Row. Cos Dr. Sag.:</td>
<td>1</td>
</tr>
<tr>
<td>Im. Row. Cos Dr. Cor.:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Row. Cos Dr. Tra.:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Col. Cos Dr. Sag.:</td>
<td>0</td>
</tr>
<tr>
<td>Im. Col. Cos Dr. Cor.:</td>
<td>1</td>
</tr>
<tr>
<td>Im. Col. Cos Dr. Tra.:</td>
<td>0</td>
</tr>
<tr>
<td>Seqn. Phase Cor. Rows:</td>
<td>-19222</td>
</tr>
<tr>
<td>Seqn. Phase Cor. Cols.:</td>
<td>-19222</td>
</tr>
<tr>
<td>Room. Phase Cor. Rows:</td>
<td>0</td>
</tr>
<tr>
<td>Room. Phase Cor. Cols.:</td>
<td>0</td>
</tr>
<tr>
<td>Nominal 3D Raw Parts:</td>
<td>0</td>
</tr>
<tr>
<td>Current 3D Raw Parts:</td>
<td>0</td>
</tr>
<tr>
<td>Tot. 3D Raw Parts:</td>
<td>0</td>
</tr>
<tr>
<td>Actual 3D Raw Parts:</td>
<td>-19222</td>
</tr>
<tr>
<td>Nominal Total Slices:</td>
<td>5</td>
</tr>
<tr>
<td>Current Total Slices:</td>
<td>5</td>
</tr>
<tr>
<td>Current Slice Number:</td>
<td>2</td>
</tr>
<tr>
<td>Current Group Number:</td>
<td>2</td>
</tr>
<tr>
<td>MIP Start Row:</td>
<td>-19222</td>
</tr>
<tr>
<td>MIP Stop Row:</td>
<td>-19222</td>
</tr>
<tr>
<td>MIP Start Column:</td>
<td>-19222</td>
</tr>
<tr>
<td>MIP Stop Column:</td>
<td>-19222</td>
</tr>
<tr>
<td>MIP Start Slice:</td>
<td>-19222</td>
</tr>
<tr>
<td>MIP Stop Slice:</td>
<td>-19222</td>
</tr>
<tr>
<td>Signal Mask:</td>
<td>0</td>
</tr>
<tr>
<td>Trigger Delay (ms):</td>
<td>-19222</td>
</tr>
<tr>
<td>RR Interval (ms):</td>
<td>-19222</td>
</tr>
<tr>
<td>Tot. Trigger Pulses:</td>
<td>-19222</td>
</tr>
<tr>
<td>AQC Voltage (V):</td>
<td>10</td>
</tr>
<tr>
<td>Transmit. Amplt. (V):</td>
<td>769.43535551456</td>
</tr>
</tbody>
</table>
4.3.4 Viewing imported MR Images from database

Imported images can be instantly viewed by the GUI for import routine. This facility can be used to see if the processed images are properly imported. This also visualises various quantifiable image parameters. Later on these parameters can be used to search images, which is discussed in the following section.

Figure 12 – GUI for viewing importing images
4.3 Import routine

View of imported image header information from header BLOB field as specified in Schema 2.

Figure 13 – GUI for viewing imported image header information

Most important image parameters such as file name, file sequence, echo time, image maximum intensity, image height and image width are extracted from the header block and stored into database field as defined in Schema 2.

The import routine also separates the entire header data and stores them into a BLOB database field \textit{IMAGEHEADER} as specified in Schema 2. This allows for further processing by specific plug-in module.
4.4 Search and selection

4.4.1 Principle of search and selection

The search and selection routine of MR image database tool utilises standard RDBMS "SELECT" SQL command to filter out the desired images from the database and visualises with the help of purpose designed GUI.

The most important search criterion is the implementation of search facility by image parameters. While searching, the user is able to select any parameter and specify its value or a range of values, then the search routine automatically builds SELECT SQL codes and submits them to the RDBMS. RDBMS processes these SQL codes at the database server-end and return a set of images that satisfy the conditions specified within the SQL codes. Then it is the job for the GUI to interpret this data into visual format for the user to interact with.

Searched images can be visually selected and put into a group for further studies. To hold the selected images into a group, an additional two database tables are defined, (1) a group table that holds the information of the study, and (2) an image reference table that holds references to all selected images within the group. These two tables were essential to satisfy the functionality specified in the previous chapter, sections 3.3.3 and 3.3.4. The schema for creating these 2 tables is included in Appendix-B.

The detailed code for designing the search and selection routine and graphical user interface are included in Appendix-A, Search and Selection Tool.

The graphical user interface for search and selection of MR image database tool is demonstrated in Figure 14.
4.4 Search and selection

The image search and selection routine is implemented in 2 different ways: (1) as an integrated part of MRI database tool, (2) as an dynamic linking library (DLL). Both of these methods are discussed in the following 2 sections.

1. Search image database by any of the selected list of parameters for which an index is created.

2. Double click to add image to selected group.

3. Name selected group of images.

4. Return selected images to plug-in

View facility for searched images.

Figure 14 – GUI for searching, viewing and selecting images
4.4.2 Search and selection as an integrated module

The integrated image search and selection module is compiled together with the main MR image database tool. By doing so, the need for further integration with other functional modules such as plug-in module is eradicated.

The benefits of an integrated image search and selection module are:

- Clinicians can have all functional modules within one GUI and do not have to swap between too many applications.
- No need to configure ODBC to connect to the database, as it uses the same connection that is used by the main MRI database tool.
- No need to interface with an external image search and selection routine such as DLL.

The main disadvantage of an integrated image search and selection module is that it can not be used by any other application except the one it integrates with.

4.4.3 Search and selection as a DLL

To overcome the disadvantage described above, an image search and selection module is compiled as a dynamic linking library (DLL), which can be accessed by any external application that has the ability to connect to the RDBMS via ODBC or any other database connectivity tool.

A DLL, SelDBImage.DLL, is compiled to provide an image search and retrieval function GetImageSQL which is defined as following:

```pascal
function GetImageSQL(ParentWinHandle : Cardinal; var pSQL : pChar; var ImageGroupID : Integer ):Boolean ; stdcall;
```
GetImageSQL is an image search and selection routine encapsulated within SelDBImage.DLL. It provides a generic search facility for images stored within MR image database, and builds automatically a SELECT SQL query for selected images, which is then returned to host application via pSQL string pointer variable.

The search mechanism also provides a facility to group the selected images and save it under a user-defined title. For a better control, the system also assigns a unique id to this image group to avoid any duplication. This group image id can be returned to the host application via ImageGroupID (see table 6). Using this ImageGroupID the host application can reselect and investigate the selected images at a later date.

ParentWinHandle is an integer data type. Pass the parent window handle with this parameter.

pSQL is a pointer to null terminated string or array of characters. This must be initialised before passing it to the host application. pSQL will return the SQL for selected images if the selection was successful.

ImageGroupID is an integer data type. Pass ImageGroupID (see table 2) variable to retrieve the selected image group id.

GetImageSQL is a function of Boolean type. If the selection is successful, it will return a true value otherwise a false value.

---

Table 6 – image group (for T2* and M0 image) table data definition

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Constrains</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2SIMAGEID</td>
<td>INTEGER</td>
<td>NOT NULL</td>
<td>Image Group ID</td>
</tr>
<tr>
<td>T2SIMAGE</td>
<td>BLOB</td>
<td>SUB_TYPE 0 SEGMENT SIZE 80</td>
<td>Holds processed T2*</td>
</tr>
<tr>
<td>M0IMAGE</td>
<td>BLOB</td>
<td>SUB_TYPE 0 SEGMENT SIZE 80</td>
<td>Holds processed M0</td>
</tr>
<tr>
<td>T2SIMAGETITLE</td>
<td>VARCHAR(50)</td>
<td>NOT NULL</td>
<td>Image group name</td>
</tr>
</tbody>
</table>
4.5 Plug-in module

Quantitative analysis of MR images often involves complex algorithm and advanced programming which demands time, skills and manual intervention from researchers and clinicians. When this quantitative analysis is defined, it can be encapsulated into one plug-in module which can search, access and use images from MR image RDBMS to perform quantitative analysis via a simple GUI. The biggest advantage of such a plug-in module is that once is developed, tested and configured, any clinicians can use it without having the special skill or without knowing the underlying complex algorithm; clinicians will simply utilise the result yielded by the plug-in module.

To demonstrate how a plug-in module can be developed and configured, a plug-in module for calculating $T_2^*$ and $M_0$ is designed and demonstrated in this section.

4.5.1 Calculating $T_2^*$ and $M_0$

The calculation of $T_2^*$ and $M_0$ is based on the equation E-4(1). Where $I$ is the intensity of single MR image pixel.

$$I = M_0 e^{-\frac{I}{T_2}} \quad \text{E-4(1)}$$

In this project the linear regression data modelling technique (i.e. fitting data to a straight line) is used to calculate $T_2^*$ and $M_0$ images. Before source MR image data can be fitted on a straight line, the above equation E-4(1) is prepared to satisfy the rule of linear regression.

Taking natural log on both sides of Equation E-4(1), it can be expressed as equation E-4(2):

$$\log(I) = \log(M_0) - \frac{I}{T_2} \quad \text{E-4(2)}$$
By further simplification, E-4(2) can be written as:

\[ y = a + bt \]  \hspace{1cm} \text{E-4(3)}

where

\[ y = \log(I) \]  \hspace{1cm} \text{E-4(4)}
\[ a = \log(M_0) \]  \hspace{1cm} \text{E-4(5)}
\[ b = -\frac{1}{T_2} \]  \hspace{1cm} \text{E-4(6)}
Now applying the linear regression data modelling technique on the equation E-4(3), the value of $a$ and $b$ can be obtained as:

$$a = \frac{S_x S_y - S_x S_y}{\Delta} \tag{E-4(7)}$$

$$b = \frac{SS_y - S_x S_y}{\Delta} \tag{E-4(8)}$$

where

$$S = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}$$

$$S_x = \sum_{i=1}^{N} \frac{t_i}{\sigma_i^2}$$

$$S_y = \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2}$$

$$S_m = \sum_{i=1}^{N} \frac{t_i^2}{\sigma_i^2}$$

$$S_{ty} = \sum_{i=1}^{N} \frac{t_i y_i}{\sigma_i^2}$$

$$\Delta = SS_m - S^2$$

Note: $\sigma$ is the standard deviation and is supposed to be the same for all MR image data. In that case, $\sigma^2$ can be factored out of the sum and $\sigma$ does not appear in the solution for $a$ and $b$. Since this is the case, when the value of $\sigma$ is unknown, it can be assumed 1.

The values of $a$ and $b$ as shown in E-4(7) and E-4(8) is computed and calculated. The detailed calculation algorithm and source code are included in Appendix A.
Now the calculated value of \( a \) and \( b \) can be used within E-4(5) and E-4(6) to determine the value of \( T_2^* \) and \( M_0 \) as following:

\[
M_0 = a \log(a) \quad \text{E-4(9)}
\]

\[
T_2^* = -\frac{1}{b} \quad \text{E-4(10)}
\]

Once all \( T_2^* \) and \( M_0 \) values are calculated for a given set of selected MR images, they are assembled into images which are displayed on the screen and stored into MR image database table *Image Group* as defined in Table 6.
4.5.2 Storing $T_2^*$ and $M_0$ images in MRI database

The storage structure to hold calculated $T_2^*$ and $M_0$ images is described earlier, see section 4.4.3, Table 6 ($T2SIMAGE$, $M0IMAGE$). Both $T_2^*$ and $M_0$ images are stored into MR image database as BLOB fields. These calculated images are linked with the source images, which are grouped under a study name. The basic GUI functionalities are designed to enable users to calculate, store and view $T_2^*$ and $M_0$ and recalculate them in the future if needed.

The GUI functions for calculating $T_2^*$ and $M_0$ are demonstrated in Figure 15 – 16.

The implementation source codes for functions and GUI for calculating $T_2^*$ and $M_0$ images are included in Appendix A.
4.5 Plug-in module

(1) Calculated T2* image.

(2) Selected images.

(3) Calculate/recalculate T2* and M0 images.

Figure 15 – GUI for calculating $T_2^*$ and $M_0$ images

Here the calculated $T_2^*$ image is shown.
Figure 16 – GUI for calculating $T_2^*$ and $M_0$ images

Here the calculated $M_0$ image is shown.
CHAPTER - 5

Conclusion and future work

The aim of this research was two-fold:

(1) Develop a framework for constructing an MR image database tool to overcome a number of major problems which are faced by clinicians and researchers today, such as:
   a. automating the management of a large number of MR image files that are easily produced for various studies and manually managed in a directory structure today;
   b. eradicating repetitive work for processing and displaying images on screen for examination;
   c. searching and selecting images that are relevant to a particular study, for further analysis but without duplicating original images;
   d. encapsulating and automating the procedure for performing certain calculations on MR image parameters, and producing calculated images, and retaining the result image for future use;

(2) Research, design, develop and demonstrate a prototype of MR image database manipulation tool based on this framework to find practical solutions to the above problems.

In chapter 3 a thorough analysis of the requirement was documented, and a detailed framework for proposed MR Image Database Manipulation Tool is specified. A prototype of MR image database manipulation tool based on this framework was designed, developed, tested and demonstrated in Chapter 4. The MR image database manipulation tool designed and presented in this thesis was implemented on a clinical 1.5 T whole-body MR scanner, Siemens Magnetom Vision and tested.
The image import routine presented in this thesis challenges the conventional manual filing and management of MR images within a directory structure, as import routine can process, store and index 1541 raw MR images within 229 seconds (tested on a workstation with Windows XP Pro, 1GB RAM and 2.0GHz CPU speed). The ability to separate image parameters automatically and format MR images before storing them in the database so that they can be queried and viewed without further need of processing, makes import routine a powerful tool for manipulating MR images.

SQL based search and selection routine simplifies the entire process of viewing MR images within a graphical user interface (GUI), and provides very fast access to the stored images. While a single RDBMS is capable of holding millions of MR images, search and selection routine provides functionalities to filter out unwanted images and view images that are needed for a particular study. It also utilises imported image parameters and provides functionality to search and filter images by these parameters.

The implementation of plug-in module for quantitative image analysis opens up an entire new area for clinicians and physicists to encapsulate the complex image analytical methods into a function that uses the images stored in the database. Once the plug-in module is designed and compiled, it can be used by clinicians who do not need to know the physics behind the calculation, as the calculated image and parameters will help them to diagnose the problem or reach the conclusion. The plug-in module presented in this thesis for calculating $T_2^*$ and $M_0$ only demonstrates the power of the overall image manipulation tool within database environment. Because there is no restriction on how a plug-in module should be designed, what images it can use and which programming language should be used to develop it in, it can be adapted to perform a particular calculation on any given set of images and output the results. This makes knowledge sharing in a clinical environment a reality.
Recommendation for future work

The designed prototype of MR image database tool is only tested with experimental MR images. So it has to be implemented in a clinical environment to carry out further tests and fine-tune its functionalities.

The MR image database tool specified database connectivity with electronic patient record system, which needs further integration work to implement it.

The prototype of MR image database tool presented in this thesis manipulates two dimensional static MR images, so there is a scope for further work to design tools for manipulating three dimensional real-time images or constructing virtual reality from three dimensional MR images.

Most importantly, the further development on plug-in modules for performing quantitative image analysis can lead to developing automated MR tools that can diagnose life-threatening diseases such as cancer and multiple sclerosis at their very early stages.
References


[9] Polina Golland, Spatial Encoding In MRI And How To Make It Faster, Massachusetts Institute of Technology (MIT), May 10, 2000.


[15] Dr. Simon J. Doran, Medical Physics MSc NMR and MRI, Department of Physics, University of Surrey, UK.


[51] On-line Medical Dictionary, http://cancerweb.ncl.ac.uk/cgi-bin/omd


[71] InterBase Operation guide, Borland Inc. http://www.borland.com/
[72] Database Performance Tuning and Optimization, by Sitansu S. Mittra
[73] Database Performance Tuning Handbook, by Jeff Dunham
[74] SQL Performance Tuning, by Peter Gulutzan, Trudy Pelzer
[79] Alexander Horasca,b,T, Ricardo Blana, Dorie Eigenmann, EFMI reference image database initiative:Concept, state and related work, Department of Medical Statistics and Epidemiology, Munich University of Technology, Munich, Germany, and Department of Computer Science, University of Tromsø, Tromsø, Norway
Bibliography


Appendix – A, GUI Source code

Main image processing application

Programming module

unit form_JPMmain;

interface

uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Menus, ComCtrls, ExtCtrls;

type

TArgVarray[0..2] of Pointer;
PArgV=*TArgV;

TformJPMmain = class(TForm)

MainMenu1: TMainMenu;
System1: TMenultem;
Exit1: TMenultem;
Images1: TMenultem;
Help1: TMenultem;
About1: TMenultem;
HowtoUseHelp1: TMenultem;
SearchforHelpOn1: TMenultem;
Contents1: TMenultem;
N1: TMenultem;
N2: TMenultem;
Importrawimagetodatabase1: TMenultem;
Timer1: TTimer;
StatusBar1: TStatusBar;
Viewimagesfromdatabase1: TMenultem;
Window1: TMenultem;
ArrangeAll1: TMenultem;
Cascadel: TMenultem;
file1: TMenultem;
Pluglns1: TMenultem;
Register1: TMenultem;
N3: TMenultem;
LoadIDL1: TMenultem;
N4: TMenultem;
T2Simage1: TMenultem;
N5: TMenultem;
N6: TMenultem;
SearchImages1: TMenultem;
procedure ExitClick(Sender: TObject);

procedure Readrawimageheader1Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Importrawimagetodatabase1Click(Sender: TObject);
procedure Viewimagesfromdatabase1Click(Sender: TObject);
procedure LoadIDL1Click(Sender: TObject);
procedure T2SimageClick(Sender: TObject);
procedure SearchImages1Click(Sender: TObject);

private

function WindowExists(winName : string):boolean;

function WindowExistsA(winCaption : string): TForm;

function GetlmageSQL(ParentWinHandie : THandle; var sSQL : pchar; var imageGroupID : Integer }):boolean ;  external 'SelDBimage.dll';

function GetlmageSQL(var argc :  integer; var argv : PArgV):boolean ;  external 'SelDBImage.dll';

var

formJPMmain: TForm;

implementation

uses form_ReadRawimage, form_JImportImages,
form_ViewImages, Unit_CommonFn,
form_T2SImages, if, Unit_SelectImages;

{$R *.dfm}

function TformJPMmain.WindowExists(winName : string):boolean;

var

i : integer;
found : boolean;

begin

i:=0;
found:=false;
while not found and (i < self.MDIChildCount) do

begin

found:=(UpperCase(self.MDIchildren[i].name)=UpperCase(winName));
inc(i);
end;
result:=found;
end;

function TformJPMmain.WindowExistsA(winCaption : string): TForm;

var

i : integer;
found : boolean;

begin

i:=0;
found:=false;
while not found and (i < self.MDIChildCount) do

begin

found:=(UpperCase(self.MDIchildren[i].name)=UpperCase(winCaption));
inc(i);
end;
result:=found;
end;
while not found and (i < self.MDIChildren.Count) do
begin
  found:= (self.MDIChildren[i].Caption = winCaption);
  inc(i):
end;
if found then result:=MDIChildren[i-1];
end;

procedure TForm1Main.ExitClick(Sender: TObject);
begin
  close;
end;

procedure TForm1Main.ReadRawImageHeader1Click(Sender: TObject);
begin
  if not WindowExists('formReadRawImage') then begin
    //create form
    formReadRawImage:=TFormReadRawImage.Create(self);
    formReadRawImage.FormStyle:=fsMDICard;
    with formReadRawImage do begin
      //Left:=0;
      //Top:=0;
      visible:=true;
      //WindowState:=wsMaximized;
      end;
    end;
  else begin
    with formReadRawImage do begin
      //activate form
      Visible:=true;
      BringToFront;
      //WindowState:=wsMaximized;
      end;
    end;
  end;
end;

procedure TForm1Main.Timer1Timer(Sender: TObject);
begin
  Self.StatusBar1.Panei[0].Text:=FormatDateTime('c',now);
end;

procedure TForm1Main.FormCreate(Sender: TObject);
begin
  Self.StatusBar1.Panei[0].Text:=FormatDateTime('c',now);
end;

procedure TForm1Main.LoadIDLIClick(Sender: TObject);
begin
  RunExternalApp('ide.exe',
  '\e\Data\HB\UniS\idi55\IDL55\bin\bin.x86\');
end;

procedure TForm1Main.ViewImagesFromDatabase1Click(Sender: TObject);
begin
  if not WindowExists('formViewImages') then begin
    //create form
    formViewImages:=TFormViewImages.Create(self);
    with formViewImages do begin
      //Left:=0;
      //Top:=0;
      visible:=true;
      //WindowState:=wsMaximized;
      end;
    end;
  else begin
    with formViewImages do begin
      //activate form
      Visible:=true;
      BringToFront;
      //WindowState:=wsMaximized;
      end;
    end;
  end;
end;

procedure TForm1Main.ImportRawImageToDatabase1Click(Sender: TObject);
begin
  if not WindowExists('formImportImages') then begin
    //create form
    formImportImages:=TFormImportImages.Create(self);
  end;
end;
Main image processing application

```pascal
formT2Images:=TformT2Images.Create(nil);
with formT2Images do
begin
//Left:=0;
//Top:=0;
visible:=true;
//WindowState:=wsMaximized;
end;
end;
else
with formT2Images do
begin
activate form
Visible:=true;
BringToFront;
//WindowState:=wsMaximized;
end;
end;
end;
end;

procedure TFormMain.SearchImages1Click(Sender: TObject);
var argc : integer;
pHnd : PHandler; //pointer to cardinal
pSQL : PShortString; //maximum 255 char
argv : PArgV;
pimgID : PInteger;
begin
//build param
argc:=3;
new(argv);
new(pHnd):
pHnd^:=Self.Handle;
argv[0]:=pHnd;
pSQL:=AllocMem(255); //must allocate memory, maximum 255 char
argv[1]:=pSQL;
argv[2]:=pimgID;
if GetImageData(argc, argv) then
ShowMessage('SQL: ' + pSQL + '
Image group ID: ' + inttostr(pimgID));
else
ShowMessage('No SQL');
Dispose(argv);
FreeMemory(pSQL);
Dispose(pimgID);
end;
end;

Visual GUI module

object TFormMain: TFormMain
Left = 179
Top = 110
Width = 736
Height = 561
Caption = 'Image Processing'
Color = cBkFace
Font.Charset = DEFAULT_CHARSET
Font.Color = cWindowText
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = []
FormStyle = fsMDIForm
Menu = MainMenu1
OldCreateOrder = False
WindowState = wsMaximized
WindowMenu = Windowl
OnCreate = FormCreate
PixelsPerInch = 96
TextHeight = 13
object StatusBar1: TStatusBar
Left = 0
Top = 496
Width = 728
Height = 19
Panels = [item
Alignment = taCenter
Text = 'dd//mm/yyyy hh:mm:ss'
Width = 150
end,
item
Alignment = taCenter
Text = 'Copyright © UltraSoft Technologies Ltd, UK'
Width = 50
end>
end
object MainMenu1: TMainMenu
Left = 72
Top = 40
object System1: TMenuItem
Caption = 'System'
object Exit1: TMenuItem
Caption = 'Exit'
OnClick = Exit1Click
end
object Imagel : TMenuItem
Caption = 'image'
object Readrawimageheader1 : TMenuItem
Caption = 'Read image header'
OnClick = Readrawimageheader1Click
end
object N2: TMenuItem
Caption = '-'
end
object Importrawimagetodatabase1: TMenuItem
Caption = 'Import raw image to database'
```

Caption = 'Import Images'
OnClick = ImportRawImageToDatabaseClick
end

object N5: TMenuItem
Caption = '
end

object ViewImagesfromDatabase1: TMenuItem
Caption = 'View images from DB'
OnClick = ViewImagesfromDatabase1Click
end

end

object Plugins1: TMenuItem
Caption = 'Plug-Ins'
object Register1: TMenuItem
Caption = 'Register'
end

object N3: TMenuItem
Caption = '
end

object T2Image: TMenuItem
Caption = 'T2* Images'
OnClick = T2ImageClick
end

object N4: TMenuItem
Caption = '
end

object SearchImages1: TMenuItem
Caption = 'Select Images (SQL) (Test)' 
OnClick = SearchImages1Click
end

object N6: TMenuItem
Caption = '
end

object LoadIDL1: TMenuItem
Caption = 'Load IDL'
OnClick = LoadIDL1Click
end

end

object Window1: TMenuItem
Caption = '&Window'
object Title1: TMenuItem
Caption = '&Title'
end

object Cascade1: TMenuItem
Caption = '&Cascade'
end

object ArrangeAll1: TMenuItem
Caption = '&Arrange All'
end

end

object Help1: TMenuItem
Caption = '&Help'
object Contents1: TMenuItem
Caption = 'Contents'
end

object SearchforHelpOn1: TMenuItem
Caption = '&Search for Help On...
end

end
Import routine

Programming module

unit form_importimages;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, IBDatabase, DB, ComCtrls,
  ExtCtrls, Menus, IBDataset, IBStereoSet, IBDTable, IBSQuery,
  IBSUpdateSQL, DBGrids, DBGrids;

type
  TFormImportImages = class(TForm)
  OpenDialog1: TOpenDialog;
  SaveDialog1: TSaveDialog;
  db: TIBDatabase;
  tDB: TIBTransaction;
  Pane1: TPanel;
  PopupMenu1: TPopupMenu;
  tSP: TIBTransaction;
  SP: TIBStoredProc;
  tim: TIBTransaction;
  qlm: TIBQuery;
  uiM: TIBUpdateSQL;
  DataSource1: TDataSource;
  PageControl1: TPageControl;
  TabSheet1: TTabSheet;
  TabSheet2: TTabSheet;
  Pane5: TPanel;
  Pane6: TPanel;
  Label8: TLabel;
  sbSaveImportLog: TSpeedButton;
  Memo1: TMemo;
  Pane7: TPanel;
  GroupBox1: TGroupBox;
  Label2: TLabel;
  Label1: TLabel;
  sbGGetFilePath: TSpeedButton;
  GroupBox2: TGroupBox;
  Label3: TLabel;
  Label4: TLabel;
  sbSaveCSD: TSpeedButton;
  Label5: TLabel;
  Label6: TLabel;
  sbLoadCSD: TSpeedButton;
  radioButton2: TRadioButton;
  GroupBox3: TGroupBox;
  Pane8: TPanel;
  RadioButton1: TRadioButton;
  RadioButton2: TRadioButton;
  PageControl2: TPageControl;
  TabSheet3: TTabSheet;
  TabSheet4: TTabSheet;
  Splitter1: TSplitter;
  Central: TMenuItem;
  Label5: TLabel;
  Label6: TLabel;
  Label10: TLabel;
  SpeedButton1: TSpeedButton;
  qimFILEID: TIntegerField;
  qimFILENAME: TIBStringField;
  qimFILEFULLNAME: TIBStringField;
  qimIMAGE: TIBImage;
  qimFILEHEADER: TMemoField;
  qimMAXINTENSITY: TSmallintField;
  qimIMAGEMAXWIDTH: TSmallintField;
  qimIMAGEMAXHEIGHT: TSmallintField;
  qimIMAGEPIXELSIZE: TSmallintField;
  qimECHOTIME: TSmallintField;
  qimFILESEQUENCE: TIBStringField;

procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure sbLoadCSDClick(Sender: TObject);
procedure sbSaveCSDClick(Sender: TObject);
procedure sbTestCSDClick(Sender: TObject);
procedure sbGetGDCClick(Sender: TObject);
procedure sbGetImportImageClick(Sender: TObject);
procedure sbGetFilePathClick(Sender: TObject);
procedure sbSaveImportLogClick(Sender: TObject);
procedure CentralClick(Sender: TObject);
procedure Stretch1Click(Sender: TObject);

{ Private declarations }
function GetImageDetails(iFile: string; im: TMemoryStream; var IHeader: TStringList; var IMaxIntensity, iWidth, iHeight, iPixeiSize, echoTime: Smallint): boolean;

function GetNewKey(tblName: string): integer;

end;
Appendix – A, GUI Source code

TDBServer=Record
Local : Boolean;
Server : string[40];
DatabaseFile: string[100];
UserID : string[20];
Password : string[20];
end;

var
formImportImages: TFormImportImages;

implementation
uses DateUtils, Math, unit_SiemensRawImageFn;
($R* .dfm$

function TFormImportImages.GetNewKey(tblName : string) : integer;
begin
sp.UnPrepare;
sp.Active:=true;
sp.StoredProcName:='GENNEWKEY';
sp.Prepare;
sp.ParamByName(TBLNAME').asstring:=tblName;
sp.ExecProc;
result:=sp.parambyName('KEYVALUE').asinteger;
tsp.Commit;
sp.UnPrepare;
end;

procedure TFormImportImages.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end:

procedure TFormImportImages.sbSaveCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
if not OpenFileDialog1.Execute then exit;
fs:=TFileStream.Create(fnam,fmOpenRead);
try
fs.Read(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.sbGetGDBClick(Sender: TObject);
begin
With OpenFileDialog1 do
begin
Filter:='Databases,.GDB|All files|*.*';
FilterIndex:=1;
Title:='Select Database File';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
Edit2.Text:=OpenDialog1.FileName;
end;

procedure TFormImportImages.sbLoadCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
With OpenFileDialog1 do
begin
Filter:='Database login details,.DLD|All files|*.*';
FilterIndex:=1;
Title:='Select Database Login Details';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;

procedure TFormImportImages.sbSaveCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
if not OpenFileDialog1.Execute then exit;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.sbGetGDBClick(Sender: TObject);
begin
With OpenFileDialog1 do
begin
Filter:='Databases,.GDB|All files|*.*';
FilterIndex:=1;
Title:='Select Database File';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
Edit2.Text:=OpenDialog1.FileName;
end;

procedure TFormImportImages.sbLoadCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
With OpenFileDialog1 do
begin
Filter:='Database login details,.DLD|All files|*.*';
FilterIndex:=1;
Title:='Select Database Login Details';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;

procedure TFormImportImages.sbSaveCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
if not OpenFileDialog1.Execute then exit;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.sbGetGDBClick(Sender: TObject);
begin
With OpenFileDialog1 do
begin
Filter:='Databases,.GDB|All files|*.*';
FilterIndex:=1;
Title:='Select Database File';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
Edit2.Text:=OpenDialog1.FileName;
end;

procedure TFormImportImages.sbLoadCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
With OpenFileDialog1 do
begin
Filter:='Database login details,.DLD|All files|*.*';
FilterIndex:=1;
Title:='Select Database Login Details';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;

procedure TFormImportImages.sbSaveCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
if not OpenFileDialog1.Execute then exit;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.sbGetGDBClick(Sender: TObject);
begin
With OpenFileDialog1 do
begin
Filter:='Databases,.GDB|All files|*.*';
FilterIndex:=1;
Title:='Select Database File';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
Edit2.Text:=OpenDialog1.FileName;
end;

procedure TFormImportImages.sbLoadCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
With OpenFileDialog1 do
begin
Filter:='Database login details,.DLD|All files|*.*';
FilterIndex:=1;
Title:='Select Database Login Details';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;

procedure TFormImportImages.sbSaveCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
if not OpenFileDialog1.Execute then exit;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormImportImages.sbGetGDBClick(Sender: TObject);
begin
With OpenFileDialog1 do
begin
Filter:='Databases,.GDB|All files|*.*';
FilterIndex:=1;
Title:='Select Database File';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
Edit2.Text:=OpenDialog1.FileName;
end;

procedure TFormImportImages.sbLoadCSDClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
With OpenFileDialog1 do
begin
Filter:='Database login details,.DLD|All files|*.*';
FilterIndex:=1;
Title:='Select Database Login Details';
InitialDir:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;
procedure TFormlmportlmages.sbTestCDSDClick(Sender: TObject);
begin
with db do
begin
If Connected then Connected:=false;
if RadioButtonL.Checked then
DatabaseName:=trim(Edit2.Text)
else
DatabaseName:=trim(Edit1.Text)+':'+trim(Edit2.Text);
Params.Clear;
Params.Add('user_name='+Edit3.Text) ;
Params.Add('password='+Edit4.Text) ;
try
Connected:=true;
ShowMessage('Gonnction Successful !');
except
raise;
abort;
end;
end;
end;
end;

procedure TFormlmportlmages.sbGetFilePathC1ick(Sender: TObject):
var oldOpt : TOpenOptions; cnt, i : integer;
begin
with OpenDialog1 do
begin
oldOpt:=Options;
Options := [ofAliowMultiSeiect, ofFileMustExist];
Filter := 'Image files (*.ima)|*.ima|All files (*.*)!*.*';
if Edit5.Text-' then
IniialDir:=ExtractFilePath(Application.ExeName)
eise IniialDir:=Edit5.Text;
if not Execute then
begin
Options:=oldOpt;
eXit;
end;
Memo2.Clear;
Cnt:=Files.Count;
if cnt<2 then
begin
Label7.Caption:='0 image file.';
Label10.Caption:='of '+IntTostr(cnt)+' images.';
end
else
begin
Label7.Caption:=IntTostr(cnt)+' image file.';
Label10.Caption:='of '+IntTostr(cnt)+' images.';
end;
try
Connected:=true;
ShowMessage('Connection Successful !');
except
raise;
abort;
end;
end;
end;
procedure TFormlmportlmages.sbImportlmageC1ick(Sender: TObject);
begin
importlmages_ima_Files;
end;
procedure TFormlmportlmages.importlmages_ima_Files;
var cnt, i : integer; //, ImFIIeiD
imFlle, fileSeq : string; im : TMemoryStream;
ImMaxIntensity, imWidth, imHeight, imPixelSize, echoTime : Smallint;
ImHeader : TStringList;
begin
DisplayMsg('Initiating connection to database server...');
if Connected then Connected:=false;
if RadioButton1. Checked then
DatabaseName:=trim(Edit2.Text)
else
DatabaseName:=trim(Edit1.Text)+':'+trim(Edit2.Text);
Params.Clear;
Params.Add('user_name='+Edit3.Text) ;
Params.Add('password='+Edit4.Text) ;
try
Connected:=true;
ShowMessage('Connection Successful !');
except
raise;
abort;
end;
end;
end;
procedure TFormlmportlmages.sblmportlmageCiick(Sender: TObject);
begin
importlmages_ima_Files;
end;
procedure TFormlmportlmagesIMPORTlmageFiles;
var cnt, i : integer; //, ImFIIeiD
imFlle, fileSeq : string; im : TMemoryStream;
ImMaxIntensity, imWidth, imHeight, imPixelSize, echoTime : Smallint;
ImHeader : TStringList;
begin
DisplayMsg('Initiating connection to database server...');
if Connected then Connected:=false;
if RadioButton1. Checked then
DatabaseName:=trim(Edit2.Text)
else
DatabaseName:=trim(Edit1.Text)+':'+trim(Edit2.Text);
Params.Clear;
Params.Add('user_name='+Edit3.Text) ;
Params.Add('password='+Edit4.Text) ;
try
Connected:=true;
ShowMessage('Connection Successful !');
except
raise;
abort;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
Appendix - A, GUI Source code

```pascal
s:=f;
if s="" then exit:
i:=pos('.',s);
e:=copy(s,i, length(s)-i); //save extension
s:=copy(s,1,i-1); //save file without extension

i:=pos('-',s):
s1:=copy(s,1, i-1); //save part 1
s:=copy(s,i+1, length(s)-i); //save part 2
i:=pos('.',s); //save extension
s3:=copy(s,i+1, length(s)-i); //save part 3
result:=s1+'-'+formatfloat{'00',strtoint(s2)}+'-
'+formatfloat('000',strtoint(s3))+e;
end;

begin
with qlm do
try
DisplayMsg('importing image: '+imFile):
if not Active then Open;
insert;
FieldByName('FILENAME').AsString:=GetFormattedFileName(ExtractFileName(imFile));
FieldByName('FILEFULLNAME').AsString:=imFile;
qlmIMAGE.LoadFromStream(im);
try
for i:=0 to cnt-1 do //cnt-1
begin
im.Clear;
imHeader.Clear;
imFile:=Memo2.Lines.Strings[i];
labels.Caption:=inttostr(i+1);
Application.ProcessMessages;
GetImageDetails(imFile, im, imHeader, imMaxIntensity, 
imWidth, imHeight, imPixelSize, 
echoTime, fileSeq);
SavelmageDetails;
end;
finally
im.Free;
imHeader.Free;
end;
DisplayMsg(#13#10+'Writing to database
DisplayMsg('Please wait ...');
db.ApplyUpdates([qlm]);
if TDB.InTransaction then tdb.Commit;
DisplayMsg('Importing images is complete.');
DisplayMsg(FormatDateTime('c',now) )
FieldByName('FILEHEADER').AsString:=imHeader.Text;
FieldByName('IMAGEINTENSITY').AsInteger:=imMaxIntensity;
FieldByName('IMAGEHEIGHT').AsInteger:=imHeight;
FieldByName('IMAGEWIDTH').AsInteger:=imWidth;
FieldByName('IMAGEPIXELSIZE').AsInteger:=imPixelSize;
FieldByName('FLINDEX').AsInteger:=GetNewKey('ImageFiles')
FieldByName('FILEID').AsInteger:=GetNewKey('ImageFiles')
FieldByName('EchoTime').AsInteger:=echoTime;
FieldByName('FILESEQUENCE').AsString:=fileSeq;
Post:
tIm.CommitRetaining:
DisplayMsg('Image imported.');
except
DisplayMsg('Failed to import image.');
end;
end;

begin
DisplayMsg('Importing images is started.
DisplayMsg(FormatDateTime('c',now) +#13#10);
ConnectToDatabase;
cnt:=Memo2.Lines.Count;
InitDatabase;
im:=TMemoryStream.Create;
imHeader:=TStringList.Create;
end;
```

```pascal
function TFormImportImages.GetImageDetails(iFile : string; 
im : TMemoryStream;
imHeader : TStringList; 
var imMaxIntensity, imWidth, 
imHeight, imPixelSize, echoTime ;
var fileSeq :  string) : boolean;
type TColorOverlay = record
  case integer of
    1: (iVal: Integer);
    2: (b: ARRAY[1..4] OF BYTE);
end;
var im : TMemoryStream;
var imColor : TColor;
cl : TColorOverlay:
r : double;
begin
result:=false;
fname:=iFile;
if not FileExists(fname) then exit;
//load image file
bf:=TFileStream.Create(fname,fmOpenRead);
sf:=TMemoryStream.Create;
```

```pascal
end;
```
try
if GetHeaderData(bf, vtSi, 5, 769, 770)<2 then exit;
ImageHeaderDetails(bf, iHeader);

echo time 
echoTime:=trunc(GetHeaderData(bf, vtFD, 2, 545, 552));
//seq file
//seqSeq:=GetHeaderData(bf, vtSi, 5, 992, 1020);

//read image dimension
iWidth:=GetHeaderData(bf, vtSi, 5, 771, 772);
iHeight:=GetHeaderData(bf, vtSi, 5, 773, 774);

//seq file
//seqSeq:=GetHeaderData(bf, vtS, 3, 962, 1026);

//read image
cnt:=iWidth*iHeight;
//read progress display
Application.ProcessMessages;

//resize image to right dimension
tim.Width:=iWidth;
tim.Height:=iHeight;
//read progress display
Application.ProcessMessages;

//max intensity
iMaxIntensity:=G;

//load image
bf.Position:=imageStartPos;
//sf.SetSize(cnt);
//bf.ReadBuffer{sf,cnt);
//sf.Position:=0;
for i:=0 to cnt-1 do
begin
  //max intensity
  iMaxIntensity:=iPixColor;
  sf.WriteBuffer(iPixColor, iPixelSize);
end;
sf.Position:=0;
for i:=0 to cnt-1 do
begin
  sf.ReadBuffer(iPixColor, iPixelSize);
  r:=iPixColor*(255.0/iMaxIntensitly);
  if (iPixColor>255) then
begin
  iPixColor:=255;
end;
  oCL.b[1]:=iPixColor;
  oCL.b[2]:=iPixColor;
  oCL.b[3]:=iPixColor;
  oCL.b[4]:=G;
  tim.Canvas.Pixels[x,y]:=oCL;
end;
finally
  sf.Free;
  tim.Free;
end;

procedure TFormImportImages.Center1Click(Sender:
TObject);
begin
  Centre1.Checked:=not Centre1.Checked;
  DBImage1.Center:=Centre1.Checked;
end;

procedure TFormImportImages.Stretch1Click(Sender:
TObject);
begin
end;

procedure TFormImportImages.FormCreate(Sender:
TObject);
begin
  PageControl1.ActivePageIndex:=0;
end;

procedure TFormImportImages.SpeedButton1Click(Sender:
TObject);
begin
  Memo1.Clear;
end;

procedure TFormImportImages.SaveImportLogClick(Sender:
TObject);
begin
  var fnam : string;
  fnam:=ExtractFilePath(Application.ExeName);
  fnam:=fnam+'log';
  ForceDirectories(fnam);
  fnam:=fnam+'import '+FormatDateTime('yyymmdd-
hhmmss', now)+'.log';
  Memo1.Lines.SaveToFile(fnam);
end;
Appendix -- A, GUI Source code

TValueType=(vtSI, vtLI, vtFS, vtFD, vtS); {small int, long int, float double, float single, string}

 TFormReadRawImage = class(TForm) {Main form object}
  ImagesList: TImageList;
  TreeView1: TTreeView;
  StatusBar1: TStatusBar;
  PageControl: TPageControl;
  TabSheet1: TTabSheet;
  TabSheet2: TTabSheet;
  Splitter1: TSplitter;
  Timer1: TTimer;
  Panel1: TPanel;
  Menubar1: TMenu;
  OpenDialog1: TOpenDialog;
  Panel3: TPanel;
  ToolBar1: TToolBar;
  ToolButton1: TToolButton;
  ToolButton2: TToolButton;
  ToolButton3: TToolButton;
  ToolButton4: TToolButton;
  ToolButton5: TToolButton;
  SpeedButton1: TSpeedButton;
  SpeedButton2: TSpeedButton;
  SaveDialog1: TSaveDialog;
  Label1: TLabel;
  fnProg: TPPanel;
  Label2: TLabel;
  ProgressBar1: TProgressBar;
  fMMGraphicImage;
  Panel6: TPanel;
  PopupMenu1: TPopupMenu;
  Stretch1: TMenuitem;
  MainMenu: TMenu;
  IMR1: TImage;
  Label3: TLabel;
  Panel4: TPanel;
  Label4: TLabel;
  Edit1: TEdit;
  cbUseMask: TCheckBox;
  N1: TMenuitem;
  Savel: TMenuitem;
  SpeedButton3: TSpeedButton;
  Transparent1: TMenuitem;
  Edit2: TEdit;
  Edit3: TEdit;
  procedure Save1Click(Sender: TObject);
  procedure Transparent1Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
private
  { Private declarations }
  imageLoaded: boolean;
function GetAbsBufferPos(BlockNum : Smallint; RelativePos : Integer):integer;
function GetHeaderData(Stream : TFormReadRawImage File; vt : TValueType; BlockNum : Smallint; aPos, ePos : Integer): Variant;
procedure ReadSlmenHeaderInfo(ImageFile : string);
procedure ListinTreeView(ImageFile : string);
procedure ViewImage;
public
  { Public declarations }
end;

IMRIBak: TImage;

formReadRawImage: TFormReadRawImage;

implementation

uses Math;

{$R *.dfm}

function TFormReadRawImage.GetAbsBufferPos(BlockNum : Smallint; RelativePos : Integer):integer;
var dBlocl : array[1 ..6] of Integer;
begin
  dBlocl[1]:=1; dBlocl[2]:=1023; dBlocl[3]:=2047;
  dBlocl[4]:=3199; dBlocl[5]:=4223; dBlocl[6]:=5247;
  result:=dBlocl[BlockNum]+RelativePos;
end;

function TFormReadRawImage.GetHeaderData(Stream : TFormReadRawImage File; vt : TValueType; BlockNum : Smallint; aPos, ePos : Integer): Variant;
var SI: Smallint; TotalBuffer: Integer; d : Double; s : string;
begin
  cbUseMask: TCheckBox;
  N1: TMenuitem;
  Save1: TMenuitem;
  SpeedButton3: TSpeedButton;
  Transparent1: TMenuitem;
  Edit2: TEdit;
  Edit3: TEdit;
  procedure Timer1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure ToolButton1Click(Sender: TObject);
  procedure ToolButton2Click(Sender: TObject);
  procedure ToolButton3Click(Sender: TObject);
  procedure SpeedButton1Click(Sender: TObject);
  procedure ToolButton6Click(Sender: TObject);
  procedure ToolButton5Click(Sender: TObject);
  procedure ToolButton4Click(Sender: TObject);
  procedure ToolButton3Click(Sender: TObject);
  procedure ToolButton2Click(Sender: TObject);
  procedure ToolButton1Click(Sender: TObject);
  procedure StretchClick1Click(Sender: TObject);
  procedure CenterClick(Sender: TObject);
  procedure Save1Click(Sender: TObject);
  procedure Transparent1Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
private
  { Private declarations }
  imageLoaded: boolean;
function GetAbsBufferPos(BlockNum : Smallint; RelativePos : Integer):integer;
function GetHeaderData(Stream : TFormReadRawImage File; vt : TValueType; BlockNum : Smallint; aPos, ePos : Integer): Variant;
procedure ReadSlmenHeaderInfo(ImageFile : string);
procedure ListinTreeView(ImageFile : string);
procedure ViewImage;
public
  { Public declarations }
end;
Import routine

```pascal
end;
TLongIntOverlay = record
  case integer of
    1: (LVa1: Integer);
    2: (b: ARRAY[1..4] OF BYTE);
  end;
var x: integer;
dv1, dv2: TDoubleOverlay;
sv1, sv2: TSingleOverlay;
v1, v2: TLongIntOverlay;
begin
  case vt of
  vtS8: SI:=swap(SI); //smallint, Word : - 2 bytes
  or 16 bits
  vtU8: //Longint, Int : - 4 bytes or 32 bits
    begin
      lv1.LVal:=l;
      for x:=1 to 4 do lv2.b[x]:=lv1.b[5-x];
      l:=lv2.LVal;
    end;
  vtFS: //Float Single : - 4 bytes or 32 bits
    begin
      sv1.SVal:=fS;
      for x:=1 to 4 do sv2.b[x]:=sv1.b[5-x];
      fS:=sv2.SVal;
    end;
  vtFD: //Double : - 8 bytes or 64 bits
    begin
      dv1.dVal:=d;
      for x:=1 to 8 do dv2.b[x]:=dv1.b[9-x];
      d:=dv2.dVal;
    end;
  end;
end;
begin
  TotalBuffer:=ePos-sPos+1;
  fStream.Position:=GetAbsBufferPos(BiockNum,sPos);
  case vt of
  vtS8:
    begin
      fStream.ReadByte(SI,TotalBuffer);
      Swap_Endian;
      result:=SI;
      exit;
    end;
  vtU8:
    begin
      fStream.ReadByte(l,TotalBuffer);
      Swap_Endian;
      result:=l;
      exit;
    end;
  vtFS:
    begin
      fStream.ReadByte(fS,TotalBuffer);
      Swap_Endian;
      result:=fS;
      exit;
    end;
  vtFD:
    begin
      fStream.ReadByte(d,TotalBuffer);
      Swap_Endian;
      result:=d;
      exit;
    end;
  end;
end;
begin
  TotalBuffer:=ePos-sPos+1;
  fStream.Position:=GetAbsBufferPos(BiockNum,sPos);
  case vt of
  vtS8:
    begin
      fStream.ReadByte(SI,TotalBuffer);
      Swap_Endian;
      result:=SI;
      exit;
    end;
  vtU8:
    begin
      fStream.ReadByte(l,TotalBuffer);
      Swap_Endian;
      result:=l;
      exit;
    end;
  vtFS:
    begin
      fStream.ReadByte(fS,TotalBuffer);
      Swap_Endian;
      result:=fS;
      exit;
    end;
  vtFD:
    begin
      fStream.ReadByte(d,TotalBuffer);
      Swap_Endian;
      result:=d;
      exit;
    end;
  end;
end;
```

procedure TFormReadRawlmage.ReadSimensHeaderlnfo(slmageFile : string);
begin
    var fs : TFiieStream;
    try
        fs:=TFileStream.Create(slmageFile,fmOpenRead);
        try
            Yr, Mn, Dy, Hr, Mi, Se, Ms : word;
            Yr:=GetHeaderData(fs, vtLI, 1, 2, 28);
            Mn:=GetHeaderData(fs, vtLI, 1, 17, 20);
            Dy:=GetHeaderData(fs, vtLI, 1, 69, 72);
            Hr:=GetHeaderData(fs, vtLI, 1, 53, 56);
            Mi:=GetHeaderData(fs, vtLI, 1, 41, 44);
            Se:=GetHeaderData(fs, vtLI, 1, 45, 48);
            Ms:=GetHeaderData(fs, vtLI, 1, 49, 52);
            WriteToDisplay('Acquis. date, year : '+lntToStr(Yr));
            WriteToDisplay('Acquis. date, month : '+lntToStr(Mn));
            WriteToDisplay('Acquis. date, day : '+lntToStr(Dy));
            WriteToDisplay('Acquis. time, hour : '+intToStr(Hr));
            WriteToDisplay('Acquis. time, minute : '+intToStr(Mi));
            WriteToDisplay('Acquis. time, second : '+intToStr(Se));
            WriteToDisplay('Acquis. time, mil. sec. : '+intToStr(Ms));
            WriteToDisplay('Acquis. time : '+FormatDateTime('hh:mm:ss', EncodeTime(Hr,Mi,Se,Ms)));
        except
            WriteToDisplay('Error in reading header data from image file."
        end;
    except
        WriteToDisplay('Error in reading header data from image file."
    end;
end;
WriteToDisplay('Acquisition Image : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 8, 12)));
WriteToDisplay('Image : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 12, 16)));
WriteToDisplay('Image Position, x (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 16, 20)));
WriteToDisplay('Image Position, y (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 20, 24)));
WriteToDisplay('Image Position, z (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 24, 28)));
WriteToDisplay('Field View Height (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 28, 32)));
WriteToDisplay('Field View Width (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 32, 36)));
WriteToDisplay('View Direction : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 36, 40)));
WriteToDisplay('Rut Direction : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 40, 44)));
WriteToDisplay('Image Pos. Seg. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 44, 48)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 48, 52)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 52, 56)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 56, 60)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 60, 64)));
WriteToDisplay('Image Pos. Sag. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 64, 68)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 68, 72)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 72, 76)));
WriteToDisplay('Image Pos. Sag. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 76, 80)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 80, 84)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 84, 88)));
WriteToDisplay('Image Pos. Sag. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 88, 92)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 92, 96)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 96, 100)));
WriteToDisplay('Image Pos. Sag. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 100, 104)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 104, 108)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 108, 112)));
WriteToDisplay('Image Pos. Sag. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 112, 116)));
WriteToDisplay('Image Pos. Cor. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 116, 120)));
WriteToDisplay('Image Pos. Tra. (mm) : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 120, 124)));
procedure TFormReadRawlmage.Button2Click(Sender: TObject);
var s : string;
begin
  s := 'Not implemented in this version.
  "Future function" +
  'View selected image.';
  showmessage(s);
end;

procedure TFormReadRawlmage.Button3Click(Sender: TObject);
begin
  ViewImage;
end;

procedure TFormReadRawlmage.Button5Click(Sender: TObject);
begin
  self.Close;
end;

procedure TFormReadRawlmage.SpeedButton1Click(Sender: TObject);
begin
  if Memo1.Text = '' then
  begin
    ShowMessage('No image file in the list.');</n    exit;
  end;
  var frame : string;
  begin
    frame := ExtractFileName(File);   -- remaining frame is the file path
    frame := frame + '.txt';
    Memo1.Lines.SaveToFile(frame);
  end;
end;

procedure TFormReadRawlmage.SpeedButton2Click(Sender: TObject);
var frame : string;
begin
  if TreeView1.Items.Count = 0 then
  begin
    ShowMessage('No image file in the list.');</n    exit;
  end;
  if TreeView1.Selected.AbsoluteIndex = 0 then
  begin
  end;

procedure TFormReadRawlmage.Button4Click(Sender: TObject);
var s : string;
begin
  s := 'SiEMENS V3 MRI FILE READER
  READ HEADER - Use open icon (top left) to open a
  *.ima file.
  'Future function'' +
  'View selected image.';
  showmessage(s);
end;

procedure TFormReadRawlmage.Button6Click(Sender: TObject);
begin
  if TreeView1.Items.Count = 0 then
  begin
    ShowMessage('No image file in the list.');</n    exit;
  end;
  if (TreeView1.Selected.AbsoluteIndex = 0) or
     (TreeView1.Selected.Text = '') then
  begin
    ShowMessage('No image file in the list.');</n    exit;
  end;

end;
showmessage('Please select a file to view image.'); exit;
end;

if not FileExists(fnam0) then begin
  showmessage(fnam0+#13#10+'does not exist.'); exit;
end;

//load image file
bf:=TFileStream.Create(fnam0,fmOpenRead):
wf:=TFileStream.Create('e:\temp\ti2.img', fmCreate);

try
  if GetHeaderData(bf, vtSI, 5, 771, 772)<>2 then begin
    showmessage(fnam0+#13#10+'does not contain 2D image.'): exit;
  end;
  ImageLoaded:=false;
  PageControl1.ActivePageIndex:=1; //activate image preview screen
  //read image dimension
  iWidth:=GetHeaderData(bf, vtSI, 5, 771, 772);
  iHeight:=GetHeaderData(bf, vtSI, 5, 773, 774);
  cnt:=iWidth*iHeight;
  iPixelSize:=2;
  //init progress display
  pnlProg.Visible:=true;
  Application.ProcessMessages;
  //resize image to right dimension
  iMRI.Bak.Width:=iWidth;
  iMRI.Bak.Height:=iHeight;
  iRead:=0;
  //load image
  if iRead<iHeight then begin
    bf.Position:=6144;
    for i=0 to cnt-1 do begin
      iRead:=iRead+iPixelSize;
      iPixelColor:=swap(iPixelColor);
      if iPixelColor>iRead then iRead:=iPixelColor;
      wf.WriteBuffer(iPixelColor, iPixelSize);
      r:=iPixelColor*(255.0/imax):
      iPixelColor:=round(r);
      if (iPixelColor>255) then begin
        iPixelColor:=255;
      end;
      oCL.b[1]:=iPixelColor; oCL.b[2]:=iPixelColor;
      oCL.b[3]:=iPixelColor; oCL.b[4]:=0;
      iMRI.Bak[x,y]:=oCL;
    end;
    Edit3.Text:=iRead;
    iMRIStretch:=iRead;
  end;
finally
  bf.Free;
  wf.Free;
  pnlProg.Visible:=false;
end;

procedure TFormReadRawImage.Stretch1Click(Sender: TObject);
begin
  iMRI.Stretch:=not iMRI.Stretch;
end;

procedure TFormReadRawImage.Centre1Click(Sender: TObject);
begin
  iMRI.Center:=not iMRI.Center;
end;

procedure TFormReadRawImage.Save1Click(Sender: TObject);
var fname, ext : string;
begin
  if not ImageLoaded then begin
    ShowMessage('No image to save.'); exit;
  end;
  with SaveDialog1 do begin
    Title:='Save Image';
    Filter:='Bitmap files|*.bmp|All files|*.*';
    FilterIndex:=1;
    if not Execute then exit;
  end;
  fname:=SaveDialog1.FileName;
  ext:=trim(ExtractFileExt(fname));
  if ext='.' then fname:=fname+'.bmp';
  iMRI.Picture.SaveToFile(fname);
end;

procedure TFormReadRawImage.Transparent1Click(Sender: TObject);
begin
  if not SaveDialog1.Execute then exit;
  //SaveDialog1.FileName;
  //iMRI.Picture.SaveToFile(SaveDialog1.FileName);
  iMRI.Transparent:=true;
end;

procedure TFormReadRawImage.Stretch1Click(Sender: TObject);
begin
  iMRI.Stretch:=Stretch1.Checked;
end;

procedure TFormReadRawImage.Centre1Click(Sender: TObject);
begin
  Centre1.Checked:=not Centre1.Checked;
  iMRI.Center:=Centre1.Checked;
end;

procedure TFormReadRawImage.Save1Click(Sender: TObject);
begin
  with SaveDialog1 do begin
    Title:='Save Image';
    //Filter:='*.jpg,*.jpeg,*.bmp';
    FilterIndex:=1;
    if not Execute then exit;
    fname:=SaveDialog1.FileName;
    ext:=trim(ExtractFileExt(fname));
    if ext='.' then fname:=fname+'.bmp';
    iMRI.Picture.SaveToFile(fname);
  end;
end;
procedure TFormReadRawImage.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action:=caFree;
end;
end.

unit unit_SiemensRawImageFn;

Interface

uses Classes, SysUtils;

Const
  ImageStartPos=6144;

Type
  TValueType=(vtSl, vtLI, vtFS, vtFD, vtS); {small int, long int, floatDouble, floatSingle, string}

function GetAbsBufferPos(BlockNum : Smallint; RelativePos : Integer): Integer;

function GetHeaderData(fStream : TFileStream; vt : TValueType; BlockNum : Smallint; StartPos, EndPos : Integer): Variant;

procedure ImageHeaderDetails(fStream : TFileStream; HeaderDetails : TStringList);

Implementation

function GetAbsBufferPos(BlockNum : Smallint; RelativePos : Integer): Integer;
var dBlock : array[1 ..6] of Integer;
begin
  dBloek[1]:=1; dBloek[2]:=1023; dBloek[3]:=2047;
  dBloek[4]:=3199; dBloek[5]:=4223; dBloek[6]:=5247;
  result:=dBloek[BlockNum]+RelativePos;
end;

function GetHeaderData(fStream : TFileStream; vt : TValueType; BlockNum : Smallint; StartPos, EndPos : Integer): Variant;
var SI : Smallint; I, TotalBuffer : Integer; fS : Single; d : Double; i : Integer;
begin
  fStream.Position:=GetAbsBufferPos(BlockNum, StartPos);
  case vt of
    vtSl:
      begin
        SI:=fStream.ReadByte;
        if ISOdd then begin
          SI:=SI\#;
          fStream.ReadByte;
        end;
      end;
    vtLI:
      begin
        for x:=1 to 4 do
          SI:=(fS := fS<# ^ fS); //read a long int
      end;
    vtFS:
      begin
        for x:=1 to 4 do
          SI:=SI<<8;
        SI:=fS;
      end;
    vtFD:
      begin
        for x:=1 to 8 do
          SI:=(d := fS<<8);
      end;
  end;
end;

procedure Swap_Endian; //Delphi does not have
//swap_endian prog, I had to write one
begin
  case vt of
    vtDoubleOverlay = record
      case Integer of
        1: (dVal: Double);
        2: (x: ARRAY[1 .. 8] OF BYTE);
      end;
    end;
    VTSingleOverlay = record
      case Integer of
        1: (fVal: Single);
        2: (x: ARRAY[1 .. 4] OF BYTE);
      end;
    end;
    VTLongOverlay = record
      case Integer of
        1: (lVal: Integer);
        2: (x: ARRAY[1 .. 4] OF BYTE);
      end;
    end;
  end;
end;

procedure TFormReadRawImage.Button2Click(Sender: TObject);
end.
WriteToDisplay('Total Trigger Pulses : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 869, 872)));  
WriteToDisplay('Echo. Rep. Time (ms) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 873, 880)));  
WriteToDisplay('Gate Phase : ' +
IntToStr(GetHeaderData(fs, vtLl, 4, 831, 834)));  
WriteToDisplay('Gate Threshold (%) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 889, 896)));  
WriteToDisplay('Gate Ratio (%) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 897, 904)));  
WriteToDisplay('Interpolated Images : ' +
IntToStr(GetHeaderData(fs, vtLl, 4, 905, 908)));  
WriteToDisplay('Echoes : ' +
IntToStr(GetHeaderData(fs, vtLl, 4, 909, 912)));  
WriteToDisplay('Second Echo Time (ms) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 913, 920)));  
WriteToDisplay('Second Rep. Time (ms) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 921, 928)));  
WriteToDisplay('Cardiac Code : ' +
IntToStr(GetHeaderData(fs, vtLl, 4, 929, 932)));  
WriteToDisplay('Current Slice. Dist. Fact. : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 937, 944)));  
WriteToDisplay('Order of Slices : ' +
IntToStr(GetHeaderData(fs, vtLl, 4, 945, 948)));  
WriteToDisplay('Slab Thickness (mm) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 953, 960)));  

// Block 6
WriteToDisplay('Installation Name : ' +
GetHeaderData(fs, vtS, 6, 408, 434));  
WriteToDisplay('Patient Birth Date : ' +
GetHeaderData(fs, vtS, 6, 709, 720));  
WriteToDisplay('Sequence Information : ' +
GetHeaderData(fs, vtS, 6, 721, 732));  
WriteToDisplay('Saturation Regions : ' +
GetHeaderData(fs, vtS, 6, 733, 744));  
WriteToDisplay('Dataset Ident. : ' +
GetHeaderData(fs, vtS, 6, 745, 771));  

Except end;
end;
Visual GUI module

object TFormImportImages: TFormImportImages
  Left = 71
  Top = 116
  Width = 538
  Height = 539
  Caption = 'Import Images into Database'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = QSFixed
  FormStyle = fsMDIChild
  OldCreateOrder = False
  Position = poDefault
  Visible = True
  OnClose = FormClose
  OnCreate = FormCreate
  PixelsPerinch = 96
  TextHeight = 13
object Panel1: TPanel
  Left = 0
  Top = 0
  Width = 850
  Height = 17
  Align = alTop
  BevelOuter = bvNone
  TabOrder = 0
end
object PageControl1: TPageControl
  Left = 0
  Top = 17
  Width = 850
  Height = 455
  ActivePage = TabSheet1
  Align = alClient
  TabOrder = 1
object TabSheet1: TTabSheet
  Caption = 'Import Images'
object Splitter1: TSplitter
  Left = 529
  Top = 0
  Height = 467
end
object Panel4: TPanel
  Left = 532
  Top = 0
  Width = 310
  Height = 467
  Align = alClient
  BevelOuter = bvNone
  BorderWidth = 2
  TabOrder = 0
object Panel5: TPanel
  Left = 2
  Top = 2
  Width = 306
  Height = 31
  Align = alTop
  BevelOuter = bvNone
  TabOrder = 0
object Label6: TLabel
  Left = 0
  Top = 8
  Width = 49
  Height = 13
  Caption = 'Import log:'
end
object sbSaveImportLog: TSpeedButton
  Left = 72
  Top = 4
  Width = 65
  Height = 22
  Hint = 'Save import log.'
  Caption = 'Save Log'
  OnClick = sbSaveImportLogClick
end
object SpeedButton1: TSpeedButton
  Left = 144
  Top = 4
  Width = 65
  Height = 22
  Hint = 'Clear content of current log.'
  Caption = 'Clear Log'
  OnClick = SpeedButton1Click
end
object Memo1: TMemo
  Left = 2
  Top = 33
  Width = 306
  Height = 432
  Align = alClient
  BorderStyle = bsNone
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Courier'
  Font.Style = QSFixed
  ParentFont = False
  TabOrder = 1
object Panel2: TPanel
Left = 0
Top = 0
Width = 623
Height = 487
Align = alLeft
BevelOuter = bvNone
BorderWidth = 2
Caption = 'Panel2'
TabOrder = 1

object GroupBox1: TGroupBox
Left = 2
Top = 2
Width = 525
Height = 161
Align = alTop
Caption = 'Database Details'
TabOrder = 0

object Label2: TLabel
Left = 24
Top = 32
Width = 34
Height = 13
Caption = 'Server:'
end

object Label1: TLabel
Left = 24
Top = 56
Width = 49
Height = 13
Caption = 'Database:'
end

object sbGetGDB: TSpeedButton
Left = 400
Top = 56
Width = 23
Height = 22
Hint = 'Get database file.'
Caption = 'Get'
ParentShowHint = False
ShowHint = True
OnClick = sbGetGDBCClick
end

object Labels: TLabel
Left = 24
Top = 80
Width = 36
Height = 13
Caption = 'User id:'
end

object Label3: TLabel
Left = 24
Top = 104
Width = 49
Height = 13
Caption = 'Password:'
end

object sbSaveCSD: TSpeedButton
Left = 80
Top = 128
Width = 49
Height = 22
Hint = 'Save database server login details.'
Caption = 'Save'
ParentShowHint = False
ShowHint = True
OnClick = sbSaveCSDClick
end

object sbLoadCSD: TSpeedButton
Left = 128
Top = 128
Width = 49
Height = 22
Hint = 'Load database server login details.'
Caption = 'Load'
ParentShowHint = False
ShowHint = True
OnClick = sbLoadCSDClick
end

object sbTestCSD: TSpeedButton
Left = 184
Top = 128
Width = 49
Height = 22
Hints = 'Test database connection.'
Caption = 'Test'
ParentShowHint = False
ShowHint = True
OnClick = sbTestCSDClick
end

object Edit2: TEdit
Left = 80
Top = 56
Width = 313
Height = 21
BorderStyle = bsNone
TabOrder = 0
end

object Edit1: TEdit
Left = 80
Top = 32
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 1
end

object Edit3: TEdit
Left = 80
Top = 80
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 2
end

object Edit4: TEdit
120 Appendix - A, GUI Source code

Left = 60
Top = 104
Width = 145
Height = 21
BorderStyle = bsNone
PasswordChar = '$'
TabOrder = 3
end
object RadioButton1: TRadioButton
Left = 240
Top = 32
Width = 57
Height = 17
Caption = 'Local'
Checked = True
TabOrder = 4
TabStop = True
end

object RadioButton2: TRadioButton
Left = 304
Top = 32
Width = 73
Height = 17
Caption = 'Remote'
TabOrder = 5
end

object GroupBox2: TGroupBox
Left = 2
Top = 163
Width = 525
Height = 302
Align = alClient
Caption = 'Import Images'
TabOrder = 1
object Panel3: TPanel
Left = 2
Top = 15
Width = 521
Height = 50
Align = alTop
BevelOuter = bvNone
TabOrder = 0
DesignSize = (521, 50)
object Label7: TLabel
Left = 24
Top = 4
Width = 43
Height = 13
Caption = 'File path:'
end
object sbGetFilePath: TSpeedButton
Left = 400
Top = 3
Width = 23
Height = 22
Hint = 'Get image files path'
Caption = '.
ParentShowHint = False
ShowHint = True
OnClick = sbGetFilePathClick
end

object sbImportImage: TSpeedButton
Left = 432
Top = 3
Width = 81
Height = 22
Hint = 'Import Images'
Caption = 'Import Images'
ParentShowHint = False
ShowHint = True
OnClick = sbImportImageClick
end

object Label7: TLabel
Left = 24
Top = 3
Width = 81
Height = 13
Caption = '0 image file.'
end

object Label8: TLabel
Left = 160
Top = 32
Width = 55
Height = 13
Caption = 'Processing:'
end

object Label9: TLabel
Left = 246
Top = 32
Width = 57
Height = 13
Caption = '0 of 0 images.'
end

object EditS: TEdit
Left = 80
Top = 4
Width = 313
Height = 21
Import routine

```plaintext
BorderStyle = bsNone
TabOrder = 0
end
end

object Memo2: TMemo
Left = 2
Top = 65
Width = 521
Height = 235
Align = alClient
BorderStyle = bsNone
ReadOnly = True
ScrollBars = ssBoth
TabOrder = 1
end
end
end

object TabSheet2: TTabSheet
Caption = 'View imported Images'
ImageIndex = 1
object Splitter2: TSplitter
Left = 409
Top = 0
Height = 467
end

object DBGrid1: TDBGrid
Left = 0
Top = 0
Width = 409
Height = 467
Align = alLeft
BorderStyle = bsNone
DataSource = DataSource1
Options = [dgTitles, dgFooterLines, dgColumnResize, dgColLines, dgRowLines, dgTabs, dgRowSelect, dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
ReadOnly = True
TabOrder = 0
TitleFont.Charset = DEFAULT_CHARSET
TitleFont.Color = clWindowText
TitleFont.Height = -11
TitleFont.Name = 'MS Sans Serif'
TitleFont.Style = 0
Columns = <
item
Expanded = False
FieldName = 'FILEID'
Title.Caption = 'Image ID'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'MAXINTENSITY'
Title.Caption = 'Max Intensity'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'IMAGEWIDTH'
Title.Caption = 'Width'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'IMAGEHEIGHT'
Title.Caption = 'Height'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'FILENAME'
Title.Caption = 'File Name'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'ECHOTIME'
Title.Caption = 'E. Time'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'FILESEQUENCE'
Title.Caption = 'File Sequence'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'FILEFULLNAME'
Title.Caption = 'File Full Name'
Width = 40
Visible = True
end>
end

object PageControl2: TPageControl
Left = 412
Top = 0
Width = 409
Height = 467
ActivePage = TabSheet3
Align = alClient
TabOrder = 1
object TabSheet3: TTabSheet
Caption = 'Image'
object DBImage1: TDBImage
Left = 0
Top = 0
Width = 422
```
Height = 439
Align = alClient
Color = cBlack
DataField = 'IMAGE'
DataSource = DataSource1
PopupMenu = PopupMenu1
TabOrder = 0
end
end
object TabSheet4: TTabSheet
Caption = 'Header Info'
ImageIndex = 1
object DBMemo1: TDBMemo
Left = 0
Top = 0
Width = 422
Height = 439
Align = alClient
DataField = 'FILEHEADER'
DataSource = DataSource1
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Courier'
Font.Style = QS
ParentFont = False
ScrollBar = sbBoth
TabOrder = 0
end
end
end
end
object OpenDialog1: TOpenDialog
Filter = 'Databases|*.GDB|All files|*.*'
Left = 464
Top = 112
end
object SaveDialog1: TSaveDialog
DefaultExt = '.dld'
Filter = 'Database login details|*.dld|All files|*.*'
Title = 'Save Database Login Details'
Left = 459
Top = 144
end
object db: TIBDatabase
Connected = True
DatabaseName = 'E:\Data\HB\UniS\Dev\ImageProc\Data\ImageDB.GDB'
Params.Strings = (
'user_name=Sysdba'
'password=masterkey')
LoginPrompt = False
IdleTimer = 0
SQLDialect = 3
TraceFlags = []
AllowStreamedConnected = False
Left = 360
end
object tDB: TIBTransaction
Active = False
DefaultDatabase = db
AutoStopAction = saNone
Left = 360
Top = 112
end
object tSP: TIBTransaction
Active = False
DefaultDatabase = db
AutoStopAction = saNone
Left = 572
Top = 97
end
object SP: TIBStoredProc
Database = db
Transaction = tSP
Left = 572
Top = 129
end
object tlm: TIBTransaction
Active = True
DefaultDatabase = db
AutoStopAction = saNone
Left = 572
Top = 169
end
object qlm: TIBQuery
Database = db
Transaction = tlm
BufferChunks = 1000
CachedUpdates = True
SQL.Strings = [
'Select I.*'
'From ImageFiles I'
'Where I.FILEID<0;')
UpdateObject = ulM
Left = 604
Top = 169
end
object qlmFILEID: TIntegerField
FieldName = 'FILEID'
Origin = 'IMAGEFILES.FILEID'
Required = True
Import routine

object qlmFILENAME: TIBStringField
Fieldname = 'FILENAME'
Origin = 'IMAGEFILES.FILENAME'
Size = 50
end

object qlmFILEFULLNAME: TIBStringField
Fieldname = 'FILEFULLNAME'
Origin = 'IMAGEFILES.FILEFULLNAME'
Size = 60
end

object qlmIMAGE: TIBBlobField
Fieldname = 'IMAGE'
Origin = 'IMAGEFILES.IMAGE'
Size = 8
end

object qlmFILEHEADER: TIBMemoField
Fieldname = 'FILEHEADER'
Origin = 'IMAGEFILES.FILEHEADER'
BlobType = ftMemo
Size = 8
end

object qlmMAXINTENSITY: TSMallintField
Fieldname = 'MAXINTENSITY'
Origin = 'IMAGEFILES.MAXINTENSITY'
end

object qlmIMAGEWIDTH: TSMallintField
Fieldname = 'IMAGEWIDTH'
Origin = 'IMAGEFILES.IMAGEWIDTH'
end

object qlmIMAGEHEIGHT: TSMallintField
Fieldname = 'IMAGEHEIGHT'
Origin = 'IMAGEFILES.IMAGEHEIGHT'
end

object qlmIMAGEPIXELSIZE: TSMallintField
Fieldname = 'IMAGEPIXELSIZE'
Origin = 'IMAGEFILES.IMAGEPIXELSIZE'
end

object qlmECHOTIME: TSMallintField
Fieldname = 'ECHOTIME'
Origin = 'IMAGEFILES.ECHOTIME'
end

object qlmFILESEQUENCE: TIBStringField
Fieldname = 'FILESEQUENCE'
Origin = 'IMAGEFILES.FILESEQUENCE'
Size = 80
end

object ulM: TIBUpdateSQL
RefreshSQL.Strings = (
  'Select '
  'from ImageFiles '
  'where'
  ' FILEID = :FILEID')
ModifySQL.Strings = (
  'update ImageFiles'
  set
  ' FILEID = :FILEID,'
  ' FILENAME = :FILENAME,'
  ' FILEFULLNAME = :FILEFULLNAME,'
  ' IMAGE = :IMAGE,'
  ' FILEHEADER = :FILEHEADER,'
  ' MAXINTENSITY = :MAXINTENSITY,'
  ' IMAGEWIDTH = :IMAGEWIDTH,'
  ' IMAGEHEIGHT = :IMAGEHEIGHT,'
  ' IMAGEPIXELSIZE = :IMAGEPIXELSIZE,'
  ' ECHOTIME = :ECHOTIME,'
  ' FILESEQUENCE = :FILESEQUENCE'
  ' where'
  ' FILEID = :OLD_FILEID')
InsertSQL.Strings = ('insert into ImageFiles'
  ' (FILEID, FILENAME, FILEFULLNAME, IMAGE, 
  FILEHEADER, MAXINTENSITY, 
  IMAGEWIDTH, IMAGEHEIGHT, IMAGEPIXELSIZE, ECHOTIME, 
  FILESEQUENCE)
  'values'
  ' (FILEID, FILENAME, FILEFULLNAME, IMAGE, 
  FILEHEADER, MAXINTENSITY, 
  IMAGEWIDTH, IMAGEHEIGHT, IMAGEPIXELSIZE, ECHOTIME, 
  FILESEQUENCE))
DeleteSQL.Strings = ('delete from ImageFiles'
  ' where'
  ' FILEID = :OLD_FILEID')

object DataSource1: TDataSource
DataSet = qlm
Left = 636
Top = 169
end

end
Image browser and viewer

Programming module

unit form_ViewImages;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, DBGrids, Grids, DBGrids,
ExtCtrls, ComCtrls,
Menus, DB, iBCustomDataSet, iBQuery, iBDatabase;

type

TformViewImages = class(TForm)

PageControl1: TPageControl;
TabSheet1: TTabSheet;
TabSheet2: TTabSheet;
Splitter2: TSplitter;
PageControl2: TPageControl;
TabSheet3: TTabSheet;
TabSheet4: TTabSheet;
DBMemo1: TDBMemo;
GroupBox1: TGroupBox;
Label2: TLabel;
Label3: TLabel;
SpeedButton1: TSpeedButton;
Label4: TLabel;
SpeedButton2: TSpeedButton;
Label5: TLabel;
Edit2: TEdit;
Edit3: TEdit;
Edit4: TEdit;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
end;

private

{ Private declarations }

procedure FetchImages;

public

{ Public declarations }

end;

var

formViewImages: TformViewImages;

implementation

uses Math;

{$R *.dfm}

procedure TFormViewImages.FormClose(Sender: TObject; var Action: TCloseAction);
begin

procedure TFormViewImages.SpeedButton1Click(Sender: TObject);

procedure TFormViewImages.SpeedButton2Click(Sender: TObject);

procedure TFormViewImages.DBGrid1TitleClick(Column: TColumn);

end;
procedure TFormViewImages.SaveCSDClick(Sender: TObject);
var fb : TDBServer;
  fnam : string;
  fs : TFileStream;
begin
  if not SaveDialog1.Execute then exit;
  fnam:=SaveDialog1.FileName;
  if uppercase(ExtractFileExt(fnam))='DLD' then
    fnam:=fnam+'.dld';
  with fb do
  begin
    Local:=RadioButton1.Checked;
    Server:=Edit1.Text;
    DatabaseFile:=Edit2.Text;
    UserID:=Edit3.Text;
    Password:=Edit4.Text;
  end;
  fs:=TFileStream.Create(fnam,fmCreate);
  try
    fs.Write(fb,sizeof(fb));
  finally
    fs.Free;
  end;
end;

procedure TFormViewImages.LoadCSDClick(Sender: TObject);
var fb : TDBServer;
  fnam : string;
  fs : TFileStream;
begin
  if not OpenDialog1.Execute then exit;
  fnam:=OpenDialog1.FileName;
  fs:=TFileStream.Create(fnam,fmOpenRead);
  try
    fs.Read(fb,sizeof(fb));
  finally
    fs.Free;
  end;
end;

procedure TFormViewImages.SpeedButton1Click(Sender: TObject);
begin
  FetchImages;
end;

procedure TFormViewImages.FetchImages;
var sSQL, sWhr : string;
procedure PrepareWhereSQL;
var s : string;
begin
  sWhr:=" Where Upper(l.FileName) =" + cbFN1.Text + ": %"
  s:=trim(Edit5.Text);
  if s<>" " then
    sWhr:=" Where Upper(l.FileName) =" + cbFN1.Text + ": %" + uppercase(trim(Edit5.Text))
  s:=trim(Edit7.Text);
  if s<>" " then
    begin
      sWhr:=s + " Where Upper(l.FileName) =" + cbFN2.Text + ": %"
      if sWhr=" " then
        sWhr:=" Where Upper(l.FileName) =" + cbFN2.Text + ": %"
end;
  with db do
  begin
    If Connected then Connected:=false;
    if RadioButton1.Checked then
      DatabaseName:=trim(Edit2.Text)
    else
      DatabaseName:=trim(Edit1.Text)+';'+trim(Edit2.Text);
    Params.Clear;
    Params.Add('user_name='+Edit3.Text);
    Params.Add('password='+Edit4.Text);
  end;
  try
    Connected:=true;
    ShowMessage('Connection Successful!');
  except
    raise;
  end;
  if Connected then Label6.Caption:="Connected to DB."
  else Label6.Caption:="Not connected to DB."
end;
```pascal
else
  sWhere:=sWhere AND Upper(I.FileName) *"touppercase.Trim(Edit7.Text))"]*%" 
end;

s:=trim(Edit2.Text);
if s<>" then
begin
  if sWhere=" then
  else
    sWhere:=sWhere+' AND Upper(I.FileName)
      cbSF1.Text*"touppercase.Trim(Edit8.Text))"]*%" 
  end;
end;

begin
  if not db.Connected then
  begin
    ShowMessage('Please open a database and then try again');
    PrepareWhereSQL;
    sSQL:='Select I.* From ImageFiles I ' + sWhere + ' Order by '+sOrderBy;
    with qlm do
    begin
      if Active then close;
      UnPrepare;
      SQL.Clear;
      SQL.Add(sSQL);
      Prepare;
      Open;
    end;
    PageControl1.ActivePageIndex:=1;
    DBGrid1.SetFocus;
  end;
end;

procedure TFormViewImages.qlmAfterScroll(DataSet: TDataSet);
begin
  //DBMemo1.SelStart;
  //DBMemo1.SelStart:=ParamPos;
  //DBMemo1.ScrollBy(50,50);
  //SendMessage(DBMemo1.Handle,
  //DBMemo1.Perform(WM_SETFOCUS));
end;

procedure TFormViewImages.qlmBeforeScroll(DataSet: TDataSet);
begin
  //ParamPos:=DBMemo1.SelStart;
end:

procedure TFormViewImages.SpeedButton2Click(Sender: TObject);
begin
  cbFN1.ItemIndex:=0;
  cbFN2.ItemIndex:=0;
  cbSF1.ItemIndex:=0;
  cbSF2.ItemIndex:=0;
  Edit5.Text=";
  Edit6.Text=";
  Edit7.Text=";
  Edit8.Text=";
  if db.Connected then FetchImages;
end;

procedure TFormViewImages.DBGrid1TitleClick(Column: TColumn);
begin
end;
```

```
else
  sWhere:=sWhere AND Upper(I.FileName) *"touppercase.Trim(Edit7.Text))"]*%" 
end;

s:=trim(Edit2.Text);
if s<>" then
begin
  if sWhere=" then
  else
    sWhere:=sWhere+' AND Upper(I.FileName)
      cbSF1.Text*"touppercase.Trim(Edit8.Text))"]*%" 
  end;
end;

begin
  if not db.Connected then
  begin
    ShowMessage('Please open a database and then try again');
    PrepareWhereSQL;
    sSQL:='Select I.* From ImageFiles I ' + sWhere + ' Order by '+sOrderBy;
    with qlm do
    begin
      if Active then close;
      UnPrepare;
      SQL.Clear;
      SQL.Add(sSQL);
      Prepare;
      Open;
    end;
    PageControl1.ActivePageIndex:=1;
    DBGrid1.SetFocus;
  end;
end;
```

```
procedure TFormViewImages.FormCreate(Sender: TObject);
begin
  PageControl1.ActivePageIndex:=0;
  sOrderBy:='!.FileName';
  SpeedButton2Click(self);
end;

procedure TFormViewImages.Stretch1Click(Sender: TObject);
begin
end;

procedure TFormViewImages.CentrelClick(Sender: TObject);
begin
  Centre1.Checked:=not Centre1.Checked;
  DBImage1.Center:=Centre1.Checked;
end;

procedure TFormViewImages.qlmAfterScroll(DataSet: TDataSet);
begin
  //DBMemo1.SelStart;
  //DBMemo1.SelStart:=ParamPos;
  //DBMemo1.ScrollBy(50,50);
  //SendMessage(DBMemo1.Handle,
  //DBMemo1.Perform(WM_SETFOCUS));
end;

end;
```

```
procedure TFormViewImages.DBGrid1TitleClick(Column: TColumn);
begin
end;
```

```
```
procedure TFormViewImages.sbGetGDBClick(Sender: TObject);
begin
  With OpenDialog1 do
  begin
    Filter := ['*.GDB*All files*.*'];
    FilterIndex := 1;
    Title := 'Select Database File';
    InitialDir := ExtractFilePath(Application.ExeName);
    end;
  if not OpenDialog1.Execute then exit;
  Edit2.Text := OpenDialog1.FileName;
end;

Visual GUI module

object TFormViewImages: TFormViewImages
  Left = 125
  Top = 149
  Width = 769
  Height = 469
  Caption = 'View images'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = QS
  FormStyle = fsMDIChild
  OldCreateOrder = False
  Position = poDefault
  Visible = True
  OnClose = FormClose
  OnCreate = FormCreate
  PixelsPerInch = 96
  TextHeight = 13

object PageControl1: TPageControl
  Left = 0
  Top = 65
  Width = 761
  Height = 377
  ActivePage = TabSheet1
  Align = alClient
  TabOrder = 0
object TabSheet1: TTabSheet
  Caption = 'Open Database'
object GroupBox1: TGroupBox
  Left = 0
  Top = 16
  Width = 441
  Caption = 'Database Details'
  TabOrder = 0
begin
  Caption := 'Server:'
end
object Label1: TLabel
  Left = 24
  Top = 32
  Width = 49
  Height = 13
  Caption := 'Database:'
end
object Label2: TLabel
  Left = 24
  Top = 56
  Width = 49
  Height = 13
  Caption := 'User id:'
end
object Label3: TLabel
  Left = 24
  Top = 80
  Width = 49
  Height = 13
  Caption := 'Password:'
end
object Label4: TLabel
  Left = 80
  Top = 128
  Width = 49
  Height = 22
  Hint := 'Save database server login details.'
  Caption := 'Save'
end
object Label5: TLabel
  Left = 400
  Top = 56
  Width = 23
  Height = 13
  Caption := 'Get database file.'
end
object Label6: TLabel
  Left = 0
  Top = 65
  Width = 761
  Height = 13
  Caption := 'Browse and viewer 127'
end
object sbSaveCSD: TSpeedButton
  Left = 80
  Top = 128
  Width = 49
  Height = 22
  Hint := 'Save database server login details.'
  Caption := 'Save'
end
object sbLoadCSD: TSpeedButton
  Left = 128
  Top = 128
  Width = 49
  Height = 22
Hint = 'Load database server login details.'
Caption = 'Load'
ParentShowHint = False
ShowHint = True
OnClick = sbLoadCSDCiick
end

object sbTestCDSD: TSpeedButton
Left = 184
Top = 128
Width = 41
Height = 22
Hint = 'Open database connection.'
Caption = 'Open'
ParentShowHint = False
ShowHint = True
OnClick = sbTestCDSDCiick
end

object Edit2: TEdit
Left = 80
Top = 56
Width = 313
Height = 21
BorderStyle = bsNone
TabOrder = 0
end

object Edit1: TEdit
Left = 80
Top = 32
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 1
end

object Edit3: TEdit
Left = 80
Top = 80
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 2
end

object Edit4: TEdit
Left = 80
Top = 104
Width = 145
Height = 21
BorderStyle = bsNone
PasswordChar = '$'
TabOrder = 3
end

object RadioButton1: TRadioButton
Left = 240
Top = 32
Width = 57
Height = 17
Caption = 'Local'
Checked = True
TabOrder = 4
end
Caption = "Not connected to DB"
Font.Charset = DEFAULT_CHARSET
Font.Color = cINavy
Font.Height = 11
Font.Name = "MS Sans Serif"
Font.Style = []
ParentFont = False
end

object Labels: TLabel
Left = 16
Top = 32
Width = 41
Height = 13
Caption = "Seq. file:"
end

object SpeedButton2: TSpeedButton
Left = 472
Top = 8
Width = 57
Height = 22
Hint = "Search images"
Caption = "Clear"
ParentShowHint = False
ShowHint = True
OnClick = SpeedButton2Click
end

object cbFNI: TComboBox
Left = 72
Top = 8
Width = 57
Height = 21
Style = csDropDownList
ItemHeight = 13
TabOrder = 2
Items.Strings = (
'Like'
'\n'>
')
end

object cbSF1: TComboBox
Left = 72
Top = 32
Width = 57
Height = 21
Style = csDropDownList
ItemHeight = 13
TabOrder = 3
Items.Strings = (
'Like'
'\n'<
')
end

object cbFN2: TComboBox
Left = 288
Top = 8
Width = 57
Height = 21
Style = csDropDownList
ItemHeight = 13
TabOrder = 4
Items.Strings = (
'Like'
'\n'<
')
end

object cbSF2: TComboBox
Left = 288
Top = 32
Width = 57
Height = 21
Style = csDropDownList
ItemHeight = 13
TabOrder = 5
Items.Strings = (
'Like'
'\n'<
')
end

object Edlist: TEdit
Left = 352
Top = 8
Width = 113
Height = 21
TabOrder = 6
OnKeyDown = Edit5KeyDown
end

object Edlist: TEdit
Left = 136
Top = 8
Width = 113
Height = 21
TabOrder = 1
OnKeyDown = Edit5KeyDown
end

object cbFN1: TComboBox
Left = 72
end
```
object EditS: TEdit
  Left = 362
  Top = 32
  Width = 113
  Height = 21
  TabOrder = 7
  OnKeyDown = EditSKeyDown
end

object db: TIBDatabase
  DatabaseName = 'E:\Data\HB\UniS\Dev\ImageProc\Data\ImageDB.GDB'
  Params.Strings = (  
    'user_name=Sysdba',  
    'password=masterkey')
  LoginPrompt = False
  IdleTimer = 0
  SQLDialect = 3
  TraceFlags = []
  AllowStreamedConnected = False
  Left = 352
  Top = 160
end

object tDB: TIBTransaction
  Active = False
  DefaultDatabase = db
  AutoStopAction = saNone
  Left = 352
  Top = 192
end

object SaveDialog1: TSaveDialog
  DefaultExt = '*.dld'
  Filter = 'Database login details|*.dld|All files|*.*'
  Title = 'Save Database Login Details'
  Left = 392
  Top = 192
end

object OpenDialog1: TOpenDialog
  Filter = 'Databases|*.GDB|All files|*.*'
  Left = 464
  Top = 112
end

object tim: TIBTransaction
  Active = False
  DefaultDatabase = db
  AutoStopAction = saNone
  Left = 572
  Top = 169
end

object qlm: TIBQuery
  Database = db
  Transaction = tim
  AfterScroll = qlmAfterScroll
  BeforeScroll = qlmBeforeScroll
  BufferChunks = 1000
  CachedUpdates = False
  ParamCheck = False
  SQL.Strings = (  
    'Select *'
    'From ImageFiles I'
    'Where I.FileID<0'
    ')
  Left = 604
  Top = 168
end

object DataSource1: TDataSource
  DataSet = qlm
  Left = 668
  Top = 169
end

object PopupMenu1: TPopupMenu
  Left = 628
  Top = 32
object Centro1: TMenuItem
  Caption = 'Centre'
  Checked = True
  OnClick = Centro1Click
end

object Stretch1: TMenuItem
  Caption = 'Stretch'
  OnClick = Stretch1Click
end
```

Search and selection tool

Programming module

```pascal
unit form_SelectImages;

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics,
    Controls, Forms,
    Dialogs, StdCtrls, Buttons, DBGrids,
    ExtCtrls, ComCtrls,
    Menus, DB, IBCustomDataSet, IBQuery, IBDatabase,
    IBUpdateSQL, Mask,
    IBStoredProc, Gauges;

type
    TFormSelectImages = class(TForm)
        PageControl1: TPageControl;
        TabSheet1: TTabSheet;
        db: TIBDatabase;
        IDB: TIBTransaction;
        SaveDialog1: TSaveDialog;
        OpenDialog1: TOpenDialog;
        tim: TIBTransaction;
        qlm: TIBQuery;
        dsim: TDataSource;
        Panel1: TPanel;
        Labels: TLabel;
        Panel2: TPanel;
        GroupBox1: TGroupBox;
        Label2: TLabel;
        Label3: TLabel;
        sbGetGDB: TSpeedButton;
        Label4: TLabel;
        Label5: TLabel;
        Label6: TLabel;
        Label7: TLabel;
        Label8: TLabel;
        Label9: TLabel;
        Label10: TLabel;
        Edit1: TEdit;
        Edit2: TEdit;
        Edit3: TEdit;
        Edit4: TEdit;
        RadioButton1: TRadioButton;
        RadioButton2: TRadioButton;
        DBGrid1: TDBGrid;
        Splitter2: TSplitter;
        PopupMenu1: TPopupMenu;
        Central1: TMenultem;
        Stretch1: TMenultem;
        PageControl2: TPageControl;
        TabSheet2: TTabSheet;
        DBImage1: TDBImage;
    private
        procedure sbSaveCSDClick(Sender: TObject);
        procedure sbLoadCSDClick(Sender: TObject);
        procedure sbTestCSDClick(Sender: TObject);
        procedure SpeedButton1Click(Sender: TObject);
        procedure Edit1KeyDown(Sender: TObject; var Key: Word;
            Shift: TShiftState);
    end;

    { Form SelectImages }

    procedure sbSaveCSDClick(Sender: TObject);
    procedure sbLoadCSDClick(Sender: TObject);
    procedure sbTestCSDClick(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure Edit1KeyDown(Sender: TObject; var Key: Word;
        Shift: TShiftState);
```

Comments:
- The code includes several classes defining the layout and functionality of the GUI.
- It uses various units such as Windows, Messages, and variants for different components.
- The interface includes tools for selecting and saving images, with options for opening and saving dialog boxes.
- It also contains methods for handling actions like saving, loading, and testing configurations.
- The source code snippet provided is part of the GUI's programming module.
procedure TFormSelectimages.sbSaveCSDClick(Sender: TObject);
  var fb : TDBServer;
  fnam : string;
  fs : TFileStream;
  begin
    with SaveDialog1 do
    begin
      Filter:='Database login details|*.DLD|All files|*.*';
      FilterIndex:=1;
      Title:=Select Database Login Details;
      InitialDir:=ExtractFilePath(Application.ExeName);
      if not Execute then exit;
      fnam:=FileName;
      fs:=TFileStream.Create(fnam, fmCreate);
      try
        fs.Write(fb,sizeof(fb));
        finally
          fs.Free;
        end;
    end;
  end;

procedure TFormSelectimages.sbLoadCSDClick(Sender: TObject);
  var fb : TDBServer;
  fnam : string;
  fs : TFileStream;
  begin
    with OpenFileDialog1 do
    begin
      Filter:='Database login details|*.DLD|All files|*.*';
      FilterIndex:=1;
      Title:=Select Database Login Details;
      InitialDir:=ExtractFilePath(Application.ExeName);
      if not Execute then exit;
      fnam:=FileName;
      fs:=TFileStream.Create(fnam, fmOpenRead);
      try
        fs.Read(fb,sizeof(fb));
        finally
          fs.Free;
        end;
    end;
    with fb do
    begin
      RadioButton1.Checked:=Local;
      RadioButton2.Checked:=not Local;
      Edit1.Text:=Server;
      Edit2.Text:=DatabaseFile;
      Edit3.Text:=UserID;
      Edit4.Text:=Password;
    end;
  end;

procedure TFormSelectimages.sbTestCSDClick(Sender: TObject);
begin
  with db do
  begin
    If Connected then Connected:=false;
    If RadioButton1.Checked then
      Server:=Edit1.Text;
      DatabaseFile:=Edit2.Text;
      UserID:=Edit3.Text;
      Password:=Edit4.Text;
      end;
    end;

end;
DatabaseName:=trim(Edit2.Text)
else
  DatabaseName:=trim(Edit1.Text)+';'+trim(Edit2.Text):
Params.Clear;
Params.Add('user_name='+Edit3.Text) ;
Params.Add('password='+Edit4.Text) ;
try
  Connected:=true;
  ShowMessage('Connection Successful!');
except
  raise:
  abort;
end;
If connected then
begin
  Label6.Caption:='Connected to OB.';
  if not qT2S.Active then qT2S.Open;
  qT2S.Last;
  DBLookupComboBox1.KeyValue:=qT2S.fieldbyName('T2S_MAGEID').asInteger;
  if not qT2SD.Active then qT2SD.Open;
end
else Label6.Caption:='Not connected to DB.';
end;

procedure TFormSelectImages.SpeedButton1Click(Sender: TObject);
begin
  FetchImages;
end;

procedure TFormSelectImages.FetchImages;
var sSQL, sWhr : string;
procedure PrepareWhereSQL;
var s : string;
begin
  sWhr:='';
  s:=trim(Edit5.Text);
  if s<>'' then
    sWhr:=Where Upper(I.FileName) '+cbFN1.Text+''*' +uppercase(s)+'' ';
  s:=trim(Edit7.Text):
  if s<>'' then
    begin
      if sWhr=''' then
        sWhr:=*Where Upper(I.FileName) '+cbFN2.Text+''*' +uppercase(s)+'' ';
      else
        sWhr:=sWhr+' AND Upper(I.FileName) '+cbFN2.Text+''*' +uppercase(s)+'' ';
    end;
  s:=trim(Edit6.Text);
  if s<>'' then
    begin
      if sWhr=''' then
        sWhr:=*Where Upper(I.FileName) '+cbFN1.Text+''*' +uppercase(s)+'' ';
      else
        sWhr:=sWhr+' AND Upper(I.FileName) '+cbFN1.Text+''*' +uppercase(s)+'' ';
    end;
  s:=trim(Edit8.Text);
  if s<>'' then
    begin
      if sWhr=''' then
        sWhr:=Where Upper(I.FileSequence) '+cbSF1.Text+''*' +uppercase(s)+'' ';
      else
        sWhr:=sWhr+' AND Upper(I.FileSequence) '+cbSF1.Text+''*' +uppercase(s)+'' ';
    end;
  s:=trim(Edit9.Text);
  if s<>'' then
    begin
      if sWhr=''' then
        sWhr:=Where I.EchoTime '+cbET.Text+s+' ';
      else
        sWhr:=sWhr+' AND I.EchoTime '+cbET.Text+s+' ';
    end;
  if sWhr=''' then sWhr= Where I.FieldID<0 ';
procedure TFormSelectImages.Edit5KeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
if Key=13 then FetchImages;
end;

function TFormSelectImages.GetNewKey(tblName : string): integer;
begin
sp.UnPrepare;
sp.Active:=true;
sp.StoredProcName:='GENNEWKEY';
sp.Prepare;
sp.ParamByName('TBLNAME').asstring:=tblName;
sp.Prepare;
sp.ExecProc;
result:=sp.parambyName('KEYVALUE').asinteger;
sp.Commit;
sp.UnPrepare;
end;

procedure TFormSelectImages.Stretch1Click(Sender: TObject);
begin
end;

procedure TFormSelectImages.Centre1Click(Sender: TObject);
begin
Centre1.Checked:=not Centre1.Checked;
DBImage1.Center:=Centre1.Checked;
end;

procedure TFormSelectImages.SpeedButton2Click(Sender: TObject);
begin
cbFN1.ItemIndex:=0;
cbFN2.ItemIndex:=0;
cbSF1.ItemIndex:=0;
cbSF2.ItemIndex:=0;
cbET.ItemIndex:=0;
Edit6.Text:=;
Edit6.Text:=;
Edit8.Text:=;
Edit9.Text:=;
if db.Connected then FetchImages;
end;

procedure TFormSelectImages.qT2SAfterInsert(DataSet: TDataSet);
begin
with DataSet do
begin
FieldByName('T2SlmageIId').AsInteger:=GetNewKey('T2Slimages');
end;
end;

function TFormSelectImages.GetNewKey(tblName : string): integer;
begin
sp.UnPrepare;
sp.Active:=true;
sp.StoredProcName:='GENNEWKEY';
sp.Prepare;
sp.ParamByName('TBLNAME').asstring:=tblName;
sp.Prepare;
sp.ExecProc;
result:=sp.parambyName('KEYVALUE').asinteger;
sp.Commit;
sp.UnPrepare;
end;

procedure TFormSelectImages.qT2SAfterPost(DataSet: TDataSet);
begin
qT2S.CommRetaining;
end;

procedure TFormSelectImages.qT2SBeforeInsert(DataSet: TDataSet);
begin
with qT2S do
begin
if not Active then //make sure the master dataset is active
begin
ShowMessage('DataSet is not active!');
abort;
end;
//make sure that master dataset has a valid entry, if not add one.
if FieldByName('T2SlmageIId').IsNull then Insert;
end;
end;

procedure TFormSelectImages.qT2SAfterInsert(DataSet: TDataSet);
begin
with DataSet do
begin
FieldByName('T2SlmageIId').AsInteger:=qT2S.FieldByName('T2SlmageIId').AsInteger;
end;
end;

procedure TFormSelectImages.DBGrid1TitleClick(Column: TColumn);
begin
sOrderBy:='.'+Column.FieldName;
if db.Connected then FetchImages;
end;

procedure TFormSelectImages.DBGrid1DblClick(Sender: TObject);
var fid : integer;
begin
with qIM do

begin if not Active then begin ShowMessage('DataSet is not active!'); abort end; if FieldByName('FileID').IsNull then begin ShowMessage('Select an image, then try again!'); abort end; end; with qT2SD do begin if not Active then begin ShowMessage('DataSet is not active!'); abort end; if not FieldByName('FileID').IsNull then fid:=FieldByName('FileID').AsInteger else fid:=0; if not (State in [dsEdit, dsInsert]) then Insert post: Close Open locate('FileID', fid, 0) end; end;

procedure TFormSelectImages(CType1Click(Sender: TObject); begin with qT2SD do begin if not Active then begin ShowMessage('DataSet is not active!'); exit end if FieldByName('T2SImageID').IsNull then begin ShowMessage('No data in DataSet!'); exit end if FieldByName('T2SImageID').IsNull then begin ShowMessage('No data in DataSet!'); exit end if FieldByName('T2SImageID').IsNull then begin ShowMessage('No data in DataSet!'); exit end end;

procedure TFormSelectImages.ctGetGDBCClick(Sender: TObject); begin With OpenFileDialog do begin Filter:='Databases|*.GDB|All fiies|*'; FilterIndex:=1 Title:='Select Database File'; InitialDir:=ExtractFilePath(Application.ExeName); end if not OpenFileDialog.Execute then exit Edit2.Text:=OpenDialog1.FileName end;

procedure TFormSelectImages.N3Click(Sender: TObject); begin with qT2SD do begin if not Active then begin ShowMessage('DataSet is not active!'); exit end; if FieldByName('T2SImageID').IsNull then begin ShowMessage('No data in DataSet!'); exit end; Edit; FieldByName('T2SImageID').Clear; FieldByName('T2SImage').Clear; Post end;

procedure TFormSelectImages.Button1Click(Sender: TObject); begin LoadFields(cbField1) end;
procedure TFormSelectImages.LoadFields(cbFields : TComboBox);
var i, cnt : integer;
begin
with qim do
begin
if not Active then
begin
ShowMessage('DataSet is not active !');
exit;
end;
cnt:=FieldDefs.Count;
cbFields.Clear;
for i:=0 to cnt-1 do
begin
if not (FieldDefs[i].DataType in [ftBlob, ftMemo, ftGraphic, ftFmtMemo, ftParadoxOle, ftDBaseOle]) then
cbFields.Items.Add(Fields[i].FieldName);
end;
end;
end;
end;
procedure TFormSelectImages.dsT2SStateChange(Sender: TObject);
begin
if dsT2S.State in [dsEdit, dsInsert] then
begin
DBLookupComboBox1.Enabled:=false;
end
else begin
DBLookupComboBox1.Enabled:=true;
end;
end;
procedure TFormSelectImages.BitBtn2Click(Sender: TObject);
var sSQL : string;
begin
sSQL:=GetSelectedSQL;
if sSQL<>" then showmessage(sSQL);
end;
function TFormSelectImages.GetSelectedSQL : String;
var sSQL: string;
begin
result:=";
with qT2S do begin
if not Active then
begin
ShowMessage('DataSet is not active !');
exit;
end;
end;
end;
end;
end.
unit form_Viewimages;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons, DBCtrls, Grids, DBGrids, ExtCtrls, Grids, Menus, DB, IBCustomDataSet, IBQuery, IBDatabase;

type
TformViewImages = class(TForm)
PageControl1: TPageControl;
TabSheet1: TTabSheet;
TabSheet2: TTabSheet;

procedure TFormViewImages.BitBtn1Click(Sender: TObject);
end;
end.
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action:=caFree;
end;

procedure TForm1.sbSaveCSDClick(Sender: TObject);
var
  fb : TDBServer;
  fnam :  string;
  fs : TFileStream;
begin
  if not SaveDialog1.Execute then exit;
  fnam:=SaveDialog1.FileName;
  if uppercase(ExtractFileExt(fnam))=" then
    fnam:=fnam+'.dld';
  with fb do
    begin
      Local:=RadioButton1.Checked;
      Server:=Edit1.Text;
      DatabaseFile:=Edit2.Text;
      UserID:=Edit3.Text;
      Password:=Edit4.Text;
    end;
  fs:=TFileStream.Create(fnam,fmCreate);
  try
    //...
procedure TFormViewImages.sbLoadCSDCIick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
With OpenFileDialog1 do
begin
Filter:=Database login details,*.DLD;All files,*.*';
FilterIndex:=1;
Title:='Select Database Login Details';
IniPath:=ExtractFilePath(Application.ExeName);
end;
if not OpenFileDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
fs:=TFileStream.Create(fnam,fmOpenRead);
try
fs.Read(fb,sizeof(fb));
finally
fs.Free;
end;
with fb do
begin
RadioButton1.Checked:=Local;
RadioButton2.Checked:=not Local;
Edit1.Text:=Server;
Edit2.Text:=DatabaseFile;
Edit3.Text:=UserID;
Edit4.Text:=Password;
end;
Connected:=true;
ShowMessage('Connection Successful!');
except
raise;
abort;
end;
IfCONNECTED then Label6.Caption:=Connected to DB.'
else Label6.Caption:=Not connected to DB.'
end;
end;
procedure TFormViewImages.SpeedButton1Click(Sender: TObject);
begin
FetchImages;
end;
procedure TFormViewImages.FetchImages;
var sSQL, sWhr : string;
procedure PrepareWhereSQL;
var s : string;
begin
sWhr;=
begin
s:=trim(Edit5.Text);
if s<>" then
begin
ifsWhr=" then
sWhr:='Where Upper(I.FileName) '+cbFN1.Text+'"+uppercase(trim(Edit5.Text))+'%''
else
sWhr:=sWhr+' AND Upper(I.FileName) '+cbFN1.Text+'"+uppercase(trim(Edit5.Text))+'%''
end;
s:=trim(Edit7.Text);
if s<>" then
begin
ifsWhr=" then
sWhr:='Where Upper(I.FileName) '+cbFN2.Text+'"+uppercase(trim(Edit7.Text))+'%''
else
sWhr:=sWhr+' AND Upper(I.FileName) '+cbFN2.Text+'"+uppercase(trim(Edit7.Text))+'%''
end;
s:=trim(Edit6.Text);
if s<>" then
begin
ifsWhr=" then
sWhr:='Where Upper(I.FileSequence) '+cbSF1.Text+'"+uppercase(trim(Edit6.Text))+'%''
else
sWhr:=sWhr+' AND Upper(I.FileSequence) '+cbSF1.Text+'"+uppercase(trim(Edit6.Text))+'%''
end;
s:=trim(Edit8.Text);
if s<>" then
begin
ifsWhr=" then
sWhr:='Where Upper(I.FileSequence) '+cbSF2.Text+'"+uppercase(trim(Edit8.Text))+'%''
else
sWhr:=sWhr+' AND Upper(I.FileSequence) '+cbSF2.Text+'"+uppercase(trim(Edit8.Text))+'%''
end;
end;
procedure TFormViewImages.sbTestCSDCIick(Sender: TObject);
begin
with db do
begin
If Connected then Connected:=false;
if RadioButton1.Checked then
DatabaseName:=trim(Edit2.Text)
else
DatabaseName:=trim(Edit1.Text)+':'+trim(Edit2.Text);
Params.Clear;
Params.Add('user_name='+Edit3.Text);
Params.Add('password='+Edit4.Text);
try
s:=trim(Edit6.Text);
if s<>" then
begin
ifsWhr=" then
sWhr:='Where Upper(I.FileSequence) '+cbSF1.Text+'"+uppercase(trim(Edit6.Text))+'%''
else
sWhr:=sWhr+' AND Upper(I.FileSequence) '+cbSF1.Text+'"+uppercase(trim(Edit6.Text))+'%''
end;
s:=trim(Edit7.Text);
if s<>" then
begin
ifsWhr=" then
sWhr:='Where Upper(I.FileName) '+cbFN2.Text+'"+uppercase(trim(Edit7.Text))+'%''
else
sWhr:=sWhr+' AND Upper(I.FileName) '+cbFN2.Text+'"+uppercase(trim(Edit7.Text))+'%''
end;
end;
procedure TFormViewImages.qlmAfterScroll(DataSet: TDataSet);
begin
  if PageControl2.ActivePage=TabSheet4 then
    begin
      //DBMemo1.SetStart;
      //DBMemo1.SetStart:=ParamPos;
      //DBMemo1.ScrollBy(0, 0);
      //SendMessage(DBMemo1.Handle,
      //DBMemo1.Perform(WM_SETFOCUS)
    end;
end;

procedure TFormViewImages.qlmBeforeScroll(DataSet: TDataSet);
begin
  //ParamPos:=DBMemo1.SelStart;
end;

procedure TFormViewImages.SpeedButton2Click(Sender: TObject);
begin
  cbFN1.ItemIndex:=0;
  cbFN2.ItemIndex:=0;
  cbSF1.ItemIndex:=0;
  cbSF2.ItemIndex:=0;
  Edit5.Text:=;
  Edit6.Text:=
  Edit7.Text:=
  if db.Connected then FetchImages;
end;

procedure TFormViewImages.DBGrid1TitleClick(Column: TColumn);
begin
  sOrderBy:='1.'+Column.FieldName;
  if db.Connected then FetchImages;
end;

procedure TFormViewImages.sbGetGDBClick(Sender: TObject);
begin
  With OpenDialog1 do
  begin
    Filter:='Databases|*.GDB|All Files|*.*';
    FilterIndex:=1;
    Title:='Select Database File';
    InitialDir:=ExtractFilePath(Application.ExeName);
    if not OpenDialog1.Execute then exit;
    Edit2.Text:=OpenDialog1.FileName;
  end;
end;
Visual GUI module

Object TFormSelectImages

Caption = 'Select Images'
Color = cBtnFace
Font.Charset = DEFAULT_CHARSET
Font.Color = ciWindowText
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = []
Icon.Data = 
OldCreateOrder = False
Position = poScreenCenter
WindowState = wsMaximized
OnCreate = FormCreate
PixelsPerlnch = 96
TextHeight = 13

Object PageControl: TPageControl

Left = 64
Top = 35
Width = 892
Height = 273
ActivePage = TabSheet1
Align = alClient
TabOrder = 0

Object TabSheet1: TTabSheet

Caption = 'Select Images'
Object Splitter2: TSplitter

Left = 625
Top = 273
Height = 257

Object Panel2: TPanel

Left = 0
Top = 0
Width = 892
Height = 273
BevelOuter = bvNone
BorderWidth = 2
TabOrder = 0

Object GroupBox1: TGroupBox

Left = 2
Top = 2
Width = 478
Height = 269
Align = alLeft
Caption = 'Open Database'
Object TabOrder = 0

Object Label2: TLabel

Left = 24
Top = 32

Width = 34
Height = 13
Caption = 'Server:'
end

object Label1: TLabel
Left = 24
Top = 56
Width = 49
Height = 13
Caption = 'Database:'
end

object SbGetGDB: TSpeedButton
Left = 460
Top = 56
Width = 23
Height = 22
Hint = 'Get database file.'
Caption = '.'
ParentShowHint = False
ShowHint = True
OnClick = SbGetGDBClick
end

object Labels: TLabel
Left = 24
Top = 80
Width = 36
Height = 13
Caption = 'User id:'
end

object Label4: TLabel
Left = 24
Top = 104
Width = 49
Height = 13
Caption = 'Password:'
end

object SbSaveCSD: TSpeedButton
Left = 50
Top = 128
Width = 49
Height = 22
Hint = 'Save database server login details.'
Caption = 'Save'
ParentShowHint = False
ShowHint = True
OnClick = SbSaveCSDClick
end

object SbLoadCSD: TSpeedButton
Left = 96
Top = 128
Width = 49
Height = 22
Hint = 'Load database server login details.'
Caption = 'Load'
ParentShowHint = False
ShowHint = True
OnClick = SbLoadCSDClick
end

object Label5: TLabel
Left = 460
Top = 128
Width = 49
Height = 22
Caption = 'File name:'
end

object SpeedButton1: TSpeedButton
Left = 350
Top = 232
Width = 67
Height = 22
Hint = 'Search images'
Caption = 'Search'
ParentShowHint = False
ShowHint = True
OnClick = SpeedButton1Click
end

object Label6: TLabel
Left = 460
Top = 208
Width = 41
Height = 13
Caption = 'Seq. file:'
end

object SpeedButton2: TSpeedButton
Left = 406
Top = 232
Width = 57
Height = 22
Hint = 'Search images'
Caption = 'Clear'
ParentShowHint = False
ShowHint = True
OnClick = SpeedButton2Click
end

object Label7: TLabel
Left = 256
Top = 188
Width = 23
Height = 13
Caption = 'AND'
end

object Label8: TLabel
Left = 256
Top = 212
Width = 23
Height = 13
Caption = 'OR'
end

object SbTestCSD: TSpeedButton
Left = 184
Top = 128
Width = 41
Height = 22
Hint = 'Open database connection.'
Caption = 'Open'
ParentShowHint = False
ShowHint = True
OnClick = SbTestCSDClick
end

object Label9: TLabel
Left = 256
Top = 212
Width = 23

BevelKind = bkFlat  
BevelOuter = bvNone  
Color = 14416381  
ItemHeight = 13  
TabOrder = 9  
Items.Strings = (  
  'Like'  
  '<'  
  ' >')  
end  
object cbFN2: TComboBox  
Left = 288  
Top = 184  
Width = 57  
Height = 21  
BevelInner = bvNone  
BevelKind = bkFlat  
BevelOuter = bvNone  
Color = 14416381  
ItemHeight = 13  
TabOrder = 10  
Items.Strings = (  
  'Like'  
  '<'  
  '>')  
end  
object cbSF2: TComboBox  
Left = 288  
Top = 208  
Width = 57  
Height = 21  
BevelInner = bvNone  
BevelKind = bkFlat  
BevelOuter = bvNone  
Color = 14416381  
ItemHeight = 13  
TabOrder = 11  
Items.Strings = (  
  '<'  
  '>'  
  '<')  
end  
object Edit7: TEdit  
Left = 350  
Top = 184  
Width = 113  
Height = 21  
BorderStyle = bsNone  
Color = 14416381  
TabOrder = 12  
OnKeyDown = EditSKeyDown  
end  
object Edit8: TEdit  
Left = 350  
Top = 208  
Width = 113  
Height = 21  
BorderStyle = bsNone  
Color = 14416381  
TabOrder = 13  
OnKeyDown = EditSKeyDown  
end  
object cbET: TComboBox  
Left = 96  
Top = 232  
Width = 57  
Height = 21  
BevelInner = bvNone  
BevelKind = bkFlat  
BevelOuter = bvNone  
Color = 14416381  
ItemHeight = 13  
TabOrder = 14  
Items.Strings = (  
  '<'  
  '>'  
  '<')  
end  
object Edit9: TEdit  
Left = 160  
Top = 232  
Width = 89  
Height = 21  
BorderStyle = bsNone  
Color = 14415381  
TabOrder = 15  
OnKeyDown = EditSKeyDown  
end  
object cbFieldl: TComboBox  
Left = 344  
Top = 144  
Width = 121  
Height = 21  
BevelInner = bvNone  
BevelKind = bkFlat  
BevelOuter = bvNone  
Color = 14416381  
ItemHeight = 13  
TabOrder = 16  
Visible = False  
Items.Strings = (  
  'Like'  
  '<'  
  ' >')  
end  
object Button1: TButton  
Left = 344  
Top = 120  
Width = 121  
Height = 21  
Caption = 'Test - Load Fields'  
TabOrder = 17  
Visible = False  
OnClick = Buttoni Click  
end  
object Edit17: TEdit  
Left = 350  
Top = 184  
Width = 113  
Height = 21  
BorderStyle = bsNone  
Color = 14416381  
TabOrder = 18  
OnKeyDown = EditSKeyDown  
end  
object Edit18: TEdit  
Left = 350  
Top = 208  
Width = 113  
Height = 21  
BorderStyle = bsNone  
Color = 14416381  
TabOrder = 19  
OnKeyDown = EditSKeyDown  
end  
object Button2: TButton  
Left = 344  
Top = 144  
Width = 121  
Height = 21  
Caption = 'Test - Load Fields'  
TabOrder = 20  
Visible = False  
OnClick = Button2Click  
end
Search and selection tool

object GroupBox2: TGroupBox
  Left = 481
  Top = 2
  Width = 409
  Height = 269
  Align = alClient
  Caption = 'Selected Images'
  TabOrder = 1

object Label1: TLabel
  Left = 8
  Top = 24
  Width = 91
  Height = 13
  Caption = 'Image group name:'
  FocusControl = DBEdit1
  Font.Charset = DEFAULT_CHARSET
  Font.Color = cIMaroon
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = [q]
  ParentFont = False
end

object DBLookupComboBox1: TDBLookupComboBox
  Left = 104
  Top = 24
  Width = 153
  Height = 21
  BevelOuter = bvNone
  KeyField = 'T2SIMAGEID'
  ListField = 'T2SIMAGETITLE'
  ListSource = dsT2S
  TabOrder = 3
end

object DBEdit1: TDBEdit
  Left = 104
  Top = 24
  Width = 137
  Height = 21
  DataField = 'T2SIMAGETITLE'
  DataSource = dsT2S
  TabOrder = 0
end

object DBNavigator1: TDBNavigator
  Left = 264
  Top = 22
  Width = 140
  Height = 25
  DataSource = dsT2S
  VisibleButtons = [nbPrior, nbNext, nbInsert, nbDelete, nbNSISI, nbReject, nbCancel]
  Flat = True
  TabOrder = 1
end

object DBGrid2: TDBGrid
  Left = 2
  Top = 56
  Width = 405
  Height = 211
  Align = alBottom
  BorderStyle = bsNone
  DataSource = dsT2S
  Options = [dgTitles, dgIndicator, dgColumnResize,
             dgColLines, dgRowLines, dgTabs, dgRowSelect,
             dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
  PopupMenu = pmT2S
  ReadOnly = True
  TabOrder = 2
  TitleFont.Charset = DEFAULT_CHARSET
  TitleFont.Color = clWindowText
  TitleFont.Height = -11
  TitleFont.Name = 'MS Sans Serif'
  TitleFont.Style = []
  OnTitleClick = DBGrid1TitleClick
  Columns = <
    Item
      Expanded = False
      FieldName = 'MAXINTENSITY'
      Title.Caption = 'Max Intensity'
      Width = 69
      Visible = True
    end
    Item
      Expanded = False
      FieldName = 'FILENAME'
      Title.Caption = 'File Name'
      Width = 90
      Visible = True
    end
    Item
      Expanded = False
      FieldName = 'ECHOTIME'
      Title.Caption = 'E Time'
      Width = 40
      Visible = True
    end
    Item
      Expanded = False
      FieldName = 'FILESEQUENCE'
      Title.Caption = 'File Sequence'
      Width = 269
      Visible = True
    end
    Item
      Expanded = False
      FieldName = 'FILEFULLNAME'
      Title.Caption = 'File Full Name'
      Width = 80
      Visible = True
    end
    Item
      Expanded = False
      FieldName = 'IMAGEWIDTH'
      Title.Caption = 'Width'
      Width = 40
      Visible = True
    end
  end
item
   Expanded = False
   FieldName = 'IMAGEHEIGHT'
   Title.Caption = 'Height'
   Width = 42
   Visible = True
end
item
   Expanded = False
   FieldName = 'FILEID'
   Title.Caption = 'Image ID'
   Width = 48
   Visible = True
end
end
end

object DBGrid1: TDBGrid
    Left = 0
    Top = 273
    Width = 625
    Height = 257
    Align = alLeft
    BorderStyle = bsNone
    DataSource = dsIm
    Options = [dgTitles, dgIndicator, dgColumnResize,
               dgColLines, dgRowLines, dgTabs, dgRowSelect,
               dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
    Readonly = True
    TabOrder = 1
    TitleFont.Charset = DEFAULT_CHARSET
    TitleFont.Color = clWindowText
    TitleFont.Height = -11
    TitleFont.Name = 'MS Sans Serif'
    TitleFont.Style = [ ]
    OnDblClick = DBGrid1DblClick
    OnTitleClick = DBGrid1TitleClick
Columns = <
   item
      Expanded = False
      FieldName = 'MAXINTENSITY'
      Title.Caption = 'Max Intensity'
      Width = 69
      Visible = True
   end
   item
      Expanded = False
      FieldName = 'FILENAME'
      Title.Caption = 'File Name'
      Width = 90
      Visible = True
   end
   item
      Expanded = False
      FieldName = 'ECHOTIME'
      Title.Caption = 'E Time'
      Width = 40
      Visible = True
   end
end

object PageControl2: TPageControl
    Left = 628
    Top = 273
    Width = 264
    Height = 257
    ActivePage = TabSheet4
    Align = alClient
    TabOrder = 2

object TabSheet3: TTabSheet
   Caption = 'Image'
   object DBImage1: TDBImage
      Left = 0
      Top = 0
      Width = 255
      Height = 229
      Align = alClient
      Color = clBlack
      DataField = 'IMAGE'
      DataSource = dsIm
      PopupMenu = PopupMenu1
      TabOrder = 0
Search and selection tool

```
end
end
object TabSheet4: TTabSheet
Caption = 'Header Info'
Imagelndex = 1
object DBMemo1: TDBMemo
Left = 0
Top = 0
Width = 256
Height = 229
Align = alClient
DataField = 'FILEHEADER'
DataSource = dsim
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Courier'
Font.Style = [q]
ParentFont = False
ScrollBars = ssBoth
TabOrder = 0
end
end
end
end
object Panel1: TPanel
Left = 0
Top = 0
Width = 900
Height = 41
Align = alTop
BevelInner = bvRaised
BevelOuter = bvNone
TabOrder = 1
object Label1: TLabel
Left = 232
Top = 13
Width = 132
Height = 13
Caption = 'Not connected to Database'
Font.Charset = DEFAULT_CHARSET
Font.Color = clNavy
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = [q]
ParentFont = False
end
object BitBtn1: TBitBtn
Left = 488
Top = 4
Width = 89
Height = 33
Caption = 'View SQL'
TabOrder = 0
OnClick = BitBtn1Click
end
```

```
```
object BitBtn3: TBitBtn
  Left = 650
  Top = 4
  Width = 75
  Height = 33
  TabOrder = 2
  Kind = bkCancel
end

object db: TIBDatabase
  Connected = True
  DatabaseName = 'E:\Data\HB\UniS\Dev\imageProc\Data\imageDB.GDB'
  Params.Strings = ('user_name=Sysdba', 'password=masterkey')
  LoginPrompt = False
  IdleTimer = 0
  SQLDiag ect = 3
  TraceFlags = Q
  AllowStreamedConnected = False
  Left = 256
  Top = 392
end

object tDB: TIBTransaction
  Active = False
  DefaultDatabase = db
  AutoStopAction = saNone
  Left = 288
  Top = 392
end

object SaveDialog1: TSaveDialog
  DefaultExt = '.did'
  Filter = 'Database login details|*.dld|All files|*.*'
  Title = 'Save Database Login Details'
  Left = 240
end

object OpenFileDialog1: TOpenDialog
  Filter = 'Databases|*.GDB|All files|*.*'
  Left = 272
  Top = 200
end

object tIm: TIBTransaction
  Active = False
  DefaultDatabase = db
  AutoStopAction = saNone
  Left = 340
  Top = 393
end

object qIm: TIBQuery
  Database = db
  Transaction = tIm
  BufferChunks = 1000
  CachedUpdates = False
  ParamCheck = False
  SQL.Strings = ('Select *', 'From ImageFiles I', 'Where I.FileID < 0')
  Left = 372
  Top = 393
end

object dsim: TDataSource
  DataSet = qIm
  Left = 404
  Top = 393
end

object PopupMenu1: TPopupMenu
  Left = 658
  Top = 396
  object Centr3: TMenultem
    Caption = 'Centre'
    Checked = True
    OnClick = Centr3 Click
  end
  object Stretc3: TMenultem
    Caption = 'Stretch'
    OnClick = Stretc3 Click
  end
end
```

Search and selection tool

AfterPost = qT2SDAfterPost
BeforeInsert = qT2SDBeforeInsert
BufferChunks = 1000
CachedUpdates = False
DataSource = dsT2S
SQL.Strings = ("Select S.T2SIMAGEID, I.*
From T2SIMAGESOURCES S inner join ImageFiles I
on S.FileID=I.FileID AND S.T2SIMageID=T2SIMageID")
UpdateObject = uT2SD
Left = 412
Top = 425
ParamData = <
item
DataType = ftInteger
Name = 'T2SIMAGEID'
ParamType = ptUnknown
Size = 4
end>
end
object dsT2SD: TDataSource
DataSet = qT2SD
Left = 444
Top = 425
end
object qT2S: TIBQuery
Database = db
Transaction = tT2S
AfterInsert = qT2SDBAfterInsert
AfterPost = qT2SDBAfterPost
BufferChunks = 1000
CachedObjects = False
SQL.Strings = ("Select T.*
From T2SIMAGES T"
'Order by T.T2SIMAGETITLE')
UpdateObject = uT2S
Left = 276
Top = 425
object qT2ST2SIMAGEID: TIntegerField
FieldName = 'T2SIMAGEID'
Origin = 'T2SIMAGES.T2SIMAGEID'
Size = 8
end
object qT2ST2SIMAGE: TBlobField
FieldName = 'T2SIMAGE'
Origin = 'T2SIMAGES.T2SIMAGE'
Size = 8
end
object qT2ST2SIMAGETITLE: TIB StringField
FieldName = 'T2SIMAGETITLE'
Origin = 'T2SIMAGES.T2SIMAGETITLE'
Size = 50
end
end
object dsT2S: TDataSource
DataSet = qT2S
OnStateChange = dsT2SStateChange
Left = 308
Top = 425
end
object uT2S: TIBUpdateSQL
RefreshSQL.Strings = {
'Select T2SIMAGEID, FILEID
From T2SIMAGESOURCES
Where T2SIMAGEID = :T2SIMAGEID and
FILEID = :FILEID'}
ModifySQL.Strings = {
'update T2SIMAGESOURCES
set T2SIMAGEID = :T2SIMAGEID,
FILEID = :FILEID
where T2SIMAGEID = :OLD_T2SIMAGEID and
FILEID = :OLD_FILEID'}
InsertSQL.Strings = {
'insert into T2SIMAGESOURCES
 (T2SIMAGEID, FILEID)
values (:T2SIMAGEID, :FILEID)'}
DeleteSQL.Strings = {
'delete from T2SIMAGESOURCES
where T2SIMAGEID = :OLD_T2SIMAGEID and
FILEID = :OLD_FILEID'}
Left = 472
Top = 424
end
end
ModifySQL.Strings = (  
'update T2SIMAGES'  
' set'  
' T2SIMAGEID = :T2SIMAGEID,'  
' T2SIMAGE = :T2SIMAGE,'  
' MOIMAGE = :MO!MAGE,'  
' T2SIMAGETITLE = :T2SIMAGETITLE'  
' where'  
' T2SIMAGEID = :OLD_T2SIMAGEID')

InsertSQL.Strings = (  
'insert into T2SIMAGES'  
' (T2SIMAGEID, T2SIMAGE, MOIMAGE, T2SIMAGETITLE)'  
'values'  
' (T2SIMAGEID, T2SIMAGE, MOIMAGE, T2SIMAGETITLE)'  
'DeleteSQL.Strings = (  
'delete from T2SIMAGES'  
' where'  
' T2SIMAGEID = :OLD_T2SIMAGEID')

end

object tSP: TIBTransaction  
Active = False  
DefaultDatabase = db  
AutoStopAction = saNone  
Left = 532  
Top = 393  
end

object SP; TIBStoredProc  
Database = db  
Transaction = tSP  
Left = 532  
Top = 425  
end

object pmT2S: TPopupMenu  
Left = 706  
Top = 220  
object Insert: TMenultem  
Caption = 'Insert'  
OnClick = InsertClck

object N1: TMenuitem  
Caption = '-'

object Deletel: TMenuitem  
Caption = 'Delete'  
OnClick = Deletel Click  
end

object N2: TMenultem  
Caption = '-'

object Cancel: TMenultem  
Caption = 'Cancel'  
OnClick = Cancel1Click

end

object FormSearchlmages: TFormSearchlmages  
Left = 36  
Top = 110  
Width = 875  
Height = 584  
Caption = 'Search images'  
Color = cIBtnFace  
Font.Charset = DEFAULT_CHARSET  
Font.Color = cWindowText  
Font.Height = -11  
Font.Name = 'MS Sans Serif'  
Font.Style = []  
OldCreateOrder = False  
PixelsPerlnch = 96  
TextHeight = 13  
object Panel3: TPanel  
Left = 0  
Top = 214  
Width = 867  
Height = 243  
Align = alClient  
BevelOuter = bvNone  
BorderWidth = 2  
Caption = PanelS  
Transaction = tSP  
DataSource = dsim

Options = [dgTitles, dgIndicator, dgColumnResize,  
dgColLines, dgRowLines, dgSort, dgFixed, dgFixedSize,  
dgFixedColSize, dgFixedRule, dgFixedRowRule, dgFixedBorder,  
dgFixedColBorder, dgFixedRowBorder, dgFixedCellBorder,  
dgAutoColumnResize, dgAutoColumnSelect,  
dgAutoColumnSort, dgAutoColumnFilter, dgAutoColumnAlignment,  
dgAutoColumnGroup, dgAutoColumnMenu, dgAutoColumnPopupMenu,  
dgAutoColumnColor, dgAutoColumnFont, dgAutoColumnStyle,  
dgAutoColumnHeight, dgAutoColumnWidth, dgAutoColumnAlignment,  
dgAutoColumnMenu, dgAutoColumnPopupMenu, dgAutoColumnColor,  
dgAutoColumnFont, dgAutoColumnStyle, dgAutoColumnHeight,  
dgAutoColumnWidth, dgAutoColumnAlignment, dgAutoColumnMenu,  
dgAutoColumnPopupMenu, dgAutoColumnColor, dgAutoColumnFont,  
dgAutoColumnStyle, dgAutoColumnHeight, dgAutoColumnWidth,  
dgAutoColumnAlignment, dgAutoColumnMenu, dgAutoColumnPopupMenu,  
dgAutoColumnColor, dgAutoColumnFont, dgAutoColumnStyle,  
dgAutoColumnHeight, dgAutoColumnWidth, dgAutoColumnAlignment,  
dgAutoColumnMenu, dgAutoColumnPopupMenu, dgAutoColumnColor,  
dgAutoColumnFont, dgAutoColumnStyle, dgAutoColumnHeight,  
dgAutoColumnWidth, dgAutoColumnAlignment, dgAutoColumnMenu,  
dgAutoColumnPopupMenu, dgAutoColumnColor, dgAutoColumnFont,  
dgAutoColumnStyle, dgAutoColumnHeight, dgAutoColumnWidth,  
dgAutoColumnAlignment, dgAutoColumnMenu, dgAutoColumnPopupMenu,  
dgAutoColumnColor, dgAutoColumnFont, dgAutoColumnStyle,  
dgAutoColumnHeight, dgAutoColumnWidth, dgAutoColumnAlignment,  
dgAutoColumnMenu, dgAutoColumnPopupMenu, dgAutoColumnColor,  
dgAutoColumnFont, dgAutoColumnStyle, dgAutoColumnHeight,  
dgAutoColumnWidth, dgAutoColumnAlignment, dgAutoColumnMenu,  
dgAutoColumnPopupMenu, dgAutoColumnColor, dgAutoColumnFont,  
dgAutoColumnStyle, dgAutoColumnHeight, dgAutoColumnWidth,  
dgAutoColumnAlignment, dgAutoColumnMenu, dgAutoColumnPopupMenu,  
dgAutoColumnColor, dgAutoColumnFont, dgAutoColumnStyle,  
dgAutoColumnHeight, dgAutoColumnWidth, dgAutoColumnAlignment,  
dgAutoColumnMenu, dgAutoColumnPopupMenu, dgAutoColumnColor,  
dgAutoColumnFont, dgAutoColumnStyle, dgAutoColumnHeight,  
dgAutoColumnWidth, dgAutoColumnAlignment, dgAutoColumnMenu,  
dgAutoColumnPopupMenu, dgAutoColumnColor, dgAutoColumnFont,  
dgAutoColumnStyle, dgAutoColumnHeight, dgAutoColumnWidth,  
dgAutoColumnAlignment, dgAutoColumnMenu, dgAutoColumnPopupMenu,  
dgAutoColumnColor, dgAutoColumnFont, dgAutoColumnStyle,  
dgAutoColumnHeight, dgAutoColumnWidth, dgAutoColumnAlignment,  
dgAutoColumnMenu, dgAutoColumnPopupMenu, dgAutoColumnColor,  
dgAutoColumnFont, dgAutoColumnStyle, dgAutoColumnHeight,  
dgAutoColumnWidth, dgAutoColumnAlignment, dgAutoColumnMenu,  
dgAutoColumnPopupMenu, dgAutoColumnColor, dgAutoColumnFont,  
dgAutoColumnStyle, dgAutoColumnHeight, dgAutoColumnWidth,  
dgAutoColumnAlignment, dgAutoColumnMenu, dgAutoColumnPopupMenu,  
dgAutoColumnColor, dgAutoColumnFont, dgAutoColumnStyle,  
dgAutoColumnHeight, dgAutoColumnWidth, dgAutoColumnAlignment,  
dgAutoColumnMenu, dgAutoColumnPopupMenu, dgAutoColumnColor,  
dgAutoColumnFont, dgAutoColumnStyle, dgAutoColumnHeight,  
dgAutoColumnWidth, dgAutoColumnAlignment, dgAutoColumnMenu,  
end

TitleFont.Charset = DEFAULT_CHARSET  
TitleFont.Color = cWindowText  
TitleFont.Height = -11  
TitleFont.Name = 'MS Sans Serif'  
TitleFont.Style = []  
Columns = <  
Item  
Expanded = False  
FieldName = 'MAXINTENSITY'  
Title.Caption = 'Max Intensity'  
Width = 68  
Visible = True
```delphi
Top = 32
Width = 34
Height = 13
Caption = "Server:"
end
object Label1: TLabel
Left = 24
Top = 56
Width = 49
Height = 13
Caption = "Database:"
end
object sbGetGDB: TSpeedButton
Left = 400
Top = 56
Width = 23
Height = 22
Hint = "Get database file."
Caption = "Get";
ParentShowHint = False
ShowHint = True
end
object Label3: TLabel
Left = 24
Top = 80
Width = 36
Height = 13
Caption = "User id:"
end
object Label4: TLabel
Left = 24
Top = 104
Width = 49
Height = 13
Caption = "Password:"
end
object sbSaveCSD: TSpeedButton
Left = 400
Top = 56
Width = 23
Height = 22
Hint = "Save database server login details."
Caption = "Save";
ParentShowHint = False
ShowHint = True
end
object Label7: TLabel
Left = 40
Top = 128
Width = 41
Height = 13
Caption = "File name:"
end
object SpeedButton1: TSpeedButton
Left = 350
Top = 232
Width = 57
Height = 22
Hint = "Search images"
Caption = "Search"
ParentShowHint = False
ShowHint = True
end
object Label9: TLabel
Left = 256
Top = 188
Width = 23
Height = 13
Caption = "AND"
end
object Label10: TLabel
Left = 256
Top = 212
Width = 23
Height = 13
Caption = "AND"
end
object Label11: TLabel
Left = 40
Top = 184
Width = 23
Height = 13
Caption = "File name:"
end
```

Appendix - A, GUI Source code
object cbFN2: TComboBox
  Left = 288
  Top = 184
  Width = 57
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 10
  Items.Strings = (
    'Like'
    '>'
    '<'
  )
end

object cbSF2: TComboBox
  Left = 288
  Top = 208
  Width = 57
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 11
  Items.Strings = (
    'Like'
    '>'
    '<'
  )
end

object Edit7: TEdit
  Left = 350
  Top = 184
  Width = 113
  Height = 21
  BorderStyle = bsNone
  Color = 14416381
  TabOrder = 12
end

object Edit8: TEdit
  Left = 350
  Top = 208
  Width = 113
  Height = 21
  BorderStyle = bsNone
  Color = 14416381
  TabOrder = 13
end

object cbETit: TComboBox
  Left = 96
  Top = 232
  Width = 57
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 14
  Items.Strings = (
    'Like'
    '>'
    '<'
  )
end

object Edit9: TEdit
  Left = 160
  Top = 232
  Width = 89
  Height = 21
  BorderStyle = bsNone
  Color = 14416381
  TabOrder = 15
end

object cbField1: TComboBox
  Left = 344
  Top = 144
  Width = 121
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 16
  Visible = False
  Items.Strings = (
    'Like'
    '>'
    '<'
  )
end

object Button1: TButton
  Left = 344
  Top = 120
  Width = 121
  Height = 25
  Caption = 'Test - Load Fields'
  TabOrder = 17
  Visible = False
end

object GroupBox2: TGroupBox
  Left = 481
  Top = 2
  Width = 384
  Height = 269
  Align = aiClient
  Caption = 'Selected images'
  TabOrder = 1
end

object Panel1: TPanel
Search and selection tool

Left = 0
Top = 0
Width = 897
Height = 41
Align = atTop
BevelInner = bvRaised
BevelOuter = bvNone
TextOrder = 0
Object Label6: TLabel
Left = 232
Top = 13
Width = 132
Height = 13
Caption = 'Not connected to Database'
Font.Charset = DEFAULT_CHARSET
Font.Color = cINavy
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = [q]
PrintedFont = False
end
end
Object db: TDatabase
Connected = True
DatabaseName = 'E:\Data\HB\UniS\Dev\imageProc\Data\imageDB.GDB'
Params.Strings = ( 'user_name=Sysdba'
'password=masterkey')
LoginPrompt = False
IdleTimer = 0
SQL dialect = 3
TraceFlags = []
AllowStreamedConnected = False
Left = 58
Top = 354
end
Object db1: TDatabase
Connected = True
DatabaseName = 'E:\Data\HB\UniS\Dev\imageProc\Data\imageDB.GDB'
Params.Strings = ( 'user_name=Sysdba'
'password=masterkey')
LoginPrompt = False
IdleTimer = 0
SQL dialect = 3
TraceFlags = []
AllowStreamedConnected = False
Left = 68
Top = 354
end
Object tDB: TTransaction
Active = False
DefaultDatabase = db
AutoStopAction = asNone
Left = 120
Top = 385
end
Object SaveDialog1: TSaveDialog
DefaultExt = '*.dld'
Filter = 'Database login details|*.dld|All files|*.*'
Title = 'Save Database Login Details'
Left = 240
Top = 200
end
Object OpenDialog1: TOpenDialog
Filter = 'Database login details|*.dld|All files|*.*'
Left = 272
Top = 200
end
Object qlm: TIBQuery
Database = db
Transaction = tT2SD
BufferClauses = 1000
CachedUpdates = False
ParamCheck = False
SQL.Strings = ( 'Select I.*
From ImageFiles I
Where I.FieldID=0' )
Left = 204
Top = 385
end
Object qT2SD: TIBQuery
Database = db
Transaction = tT2SD
BufferClauses = 1000
CachedUpdates = False
DataSource = dsT2SD
SQL.Strings = ( 'Select S.T2SimageID, I.*
From T2SIMAGESOURCES S inner join imageFiles I
on S.FileID=I.FieldID AND S.T2SimageID=T2SimageID
Order by I.FileName'
UpdateObject = uT2SD
Left = 204
Top = 497
end
object dsT2S: TDataSource
  DataSet = qT2S
  Left = 236
  Top = 497
end

object qT2S: TIBQuery
  Database = db
  Transaction = tT2S
  BufferChunks = 1000
  CachedUpdates = False
  SQL.Strings = (
    ' select T.*
    from T2SIMAGES T
    order by T.T2SIMAGETITLE'
    )
  UpdateObject = uT2S
  Left = 204
  Top = 465
end

object uT2S: TIBUpdateSQL
  RefreshSQL.Strings = (
    'Select
    ' T2SIMAGEID,' FILEID'
    ' from T2SIMAGESOURCES
    where
    ' T2SIMAGEID = :T2SIMAGEID and
    ' FILEID = :FILEID'
  )
  ModifySQL.Strings = (
    'update T2SIMAGESOURCES
    set
    ' T2SIMAGEID = :T2SIMAGEID,' FILEID = :FILEID'
    where
    ' T2SIMAGEID = :OLD_T2SIMAGEID and
    ' FILEID = :OLD_FILEID'
  )
  InsertSQL.Strings = (
    'insert into T2SIMAGESOURCES
    (T2SIMAGEID, FILEID)
    values
    (:T2SIMAGEID, :FILEID)
  )
  DeleteSQL.Strings = (
    'delete from T2SIMAGESOURCES
    where
    ' T2SIMAGEID = :OLD_T2SIMAGEID and
    ' FILEID = :OLD_FILEID'
  )
  Left = 264
  Top = 496
end

object dsT2S: TDataSource
  DataSet = qT2S
  Left = 236
  Top = 465
end
object UT2SD: TIBUpdateSQL
  RefreshSQL.Strings = (
    'Select
    ' T2SIMAGEID,' T2SIMAGE,' MOIMAGE,' T2SIMAGETITLE
    from T2SIMAGES
    where
    ' T2SIMAGEID = :T2SIMAGEID'
  )
  ModifySQL.Strings = (
    'update T2SIMAGES
    set
    ' T2SIMAGEID = :T2SIMAGEID,' T2SIMAGE = :T2SIMAGE,' MOIMAGE = :MOIMAGE,' T2SIMAGETITLE = :T2SIMAGETITLE
    where
    ' T2SIMAGEID = :OLD_T2SIMAGEID'
  )
  InsertSQL.Strings = (
    'insert into T2SIMAGES
    (T2SIMAGEID, T2SIMAGE, MOIMAGE, T2SIMAGETITLE)
    values
    (:T2SIMAGEID, :T2SIMAGE, :MOIMAGE, :T2SIMAGETITLE)
  )
  DeleteSQL.Strings = (
`delete from T2SIMAGES` where `T2SIMAGEID = :0LD_T2SIMAGEID`
T2* plug-in module

Programming module

unit form_T2Simages;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, DBControls, Grids, DBGrids,
  ExtCtrls, ComCtrls,
  Menus, DB, TCustomDataSet, TIBQuery, TIBDatabase,
  TIBUpdateSQL, Mask,
  TIBStoredProc, Gauges;

type
  TForm_T2Simages = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    db: TIBDatabase;
    tDB: TIBTransaction;
    SaveDialog1: TSaveDialog;
    OpenDialog1: TOpenDialog;
    tm: TIBTransaction;
    dbm: TIBMemo;
    cms: TDataSource;
    Panel1: TPanel;
    Label6: TLabel;
    Panel2: TPanel;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label1: TLabel;
    sbGetGDB: TSpeedButton;
    Label3: TLabel;
    sbSaveCSD: TSpeedButton;
    sbLoadCS: TSpeedButton;
    sbTestCS: TSpeedButton;
    Edit2: TEdit;
    Edit1: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    DBGrid1: TDBGrid;
    Splitter2: TSplitter;
    PopupMenu1: TPopupMenu;
    MainMenu1: TMainMenu;
    Stretch1: TMenuItem;
    PageControl2: TPageControl;
    TabSheet3: TTabSheet;
    DBImages1: TDBImage;
    TabSheet4: TTabSheet;
    DBM...
procedure ClearSearch;
public
{ Public declarations }
end;

TDBServerRecord;
Local : Boolean;
Server : string[40];
DatabaseFile: string[100];
UserID: string[20];
Password: string[20];
end;

var
formT2Slmages: TFormT2Slmages;
implementation

use Math;

{$R *.dfm}

procedure TFormT2Slmages.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Action:=caFree;
end;

procedure TFormT2Slmages.sbSaveCSCClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not SaveDialog1.Execute then exit;
fnam:=SaveDialog1.FileName;
if upperCase(ExtractFileExt(fnam))='DL' then
fnam:=fnam+'.dld';
with fb do
begin
Local:=RadioButton1.Checked;
Server:=Edit1.Text;
DatabaseFile:=Edit2.Text;
UserID:=Edit3.Text;
Password:=Edit4.Text;
end;
fs:=TFileStream.Create(fnam,fmCreate);
try
fs.Write(fb,sizeof(fb));
finally
fs.Free;
end;
end;

procedure TFormT2Slmages.sbLoadCSCClick(Sender: TObject);
var fb : TDBServer;
fnam : string;
fs : TFileStream;
begin
if not LoadDialog1.Execute then exit;
fnam:=LoadDialog1.FileName;
with fb do
begin
Local:=RadioButton1.Checked;
Server:=Edit1.Text;
DatabaseFile:=Edit2.Text;
UserID:=Edit3.Text;
Password:=Edit4.Text;
end;
end;
end;
end;
end;
With OpenDialog1 do
begin
  Filter:=Database login details (*.DLD|All files|*.*
  FilterIndex:=1;
  Title:='Select Database Login Details';
  InitialDir:=ExtractFilePath(Application.ExeName);
  end;
if not OpenDialog1.Execute then exit;
fnam:=OpenDialog1.FileName;
t:=TFileStream.Create(fnam,fmOpenRead);
try
  fs.Read(fb,sizeof(fb));
finally
  fs.Free;
end;
with fb do
begin
  RadioButton1.Checked:=Local;
  RadioButton2.Checked:=not Local;
  Edit1.Text:=Server;
  Edit2.Text:=DatabaseFile;
  Edit3.Text:=UserID;
  Edit4.Text:=Password;
end;
end;

procedure TFormT2Images.SpeedButton1Click(Sender: TObject);
begin
  FetchImages;
end;

procedure TFormT2Images.FetchImages;
var sSQL, sWhr : string;
procedure PrepareWhereSQL;
var s, s1 : string;
begin
  sWhr:='Where Upper(l.'+s1+') '+cbFN1.Text+ '"+uppercase(s)+' ';'
  if s<>" then begin
    s:=trim(Edit5.Text);
    if s<>" then begin
      sWhr:='Where Upper(l.'+s1+') '+cbFN1.Text+ '"+uppercase(s)+' ';'
    end;
  end;
end;
try
  Connected:=true;
except
  raise;
end;
if connected then begin
  Label6.Caption:='Connected to DB.';
  if not qT2S.Active then qT2S.Open;
  qT2S.First;
end;
end;

procedure TFormT2Images.SpeedButton1Click(Sender: TObject);
begin
  FetchImages;
end;

procedure TFormT2Images.FetchImages;
var sSQL, sWhr : string;
procedure PrepareWhereSQL;
var s, s1 : string;
begin
  sWhr:='Where Upper(l.'+s1+') '+cbFN1.Text+ '"+uppercase(s)+' ';'
  if s<>" then begin
    s:=trim(Edit5.Text);
    if s<>" then begin
      sWhr:='Where Upper(l.'+s1+') '+cbFN1.Text+ '"+uppercase(s)+' ';'
    end;
  end;
end;
try
  Connected:=true;
except
  raise;
end;
if connected then begin
  Label6.Caption:='Connected to DB.';
  if not qT2S.Active then qT2S.Open;
  qT2S.Close;
end;
T2* plug-in module

```
T2* plug-in module

sOrder=cbFL3.Text;
if s1<>* then begin
    s:=trim(Edit9.Text);
    if s<>* then begin
        if sWhr=' then
            sWhr:='Where Upper(l.'+S1'+') '+cbET.Text+' '+uppercase(s)+'' ';
        else
            sWhr:=sWhr+' AND Upper(l.'+S1'+') '+cbET.Text+' '+uppercase(s)+'';
    end;
end;

if sWhr=' then sWhr:='Where I.FileID<0' ;

begin
    if not db.Connected then begin
        ShowMessage('Please open a database and then try again');
        PageControl1.ActivePageIndex:=0;
    end;
    PrepareWhereSQL;
    sSQL:='Select I.* From ImageFiles I '+sWhr+' Order by '+sOrderBy;
    with qlm do begin
        if Active then close;
        UnPrepare;
        SQL.Clear;
        SQL.Add{sSQL):
        Prepare;
        Open:
        if db.Connected then FetchImages;
    end;
end;

begin
    sOrderBy:= l.'+Column.FieldName ;
    if db.Connected then FetchImages;
end;
```

```
procedure TFormT2Images.Edit5KeyDown(Sender: TObject; var Key: Word;
FieldByName('T2SlmagelD').Aslnteger:=GetNewKey('T2Images');
```

```
procedure TFormT2Images.FormCreate(Sender: TObject);
begin
    if Key=13 then FetchImages;
end;
```

```
procedure TFormT2Images.SpeedButton2Click(Sender: TObject);
begin
    ClearSearch;
end;
```

```
procedure TFormT2Images.SpeedButton2Click(Sender: TObject);
begin
    Control1.Checked:=not Control1.Checked;
    DBImage1.Center:=Control1.Checked;
end;
```

```
begin
    sOrderBy:= l.'+Column.FieldName ;
    if db.Connected then FetchImages;
end;
```

```
begin
    if Active then close;
    UnPrepare;
    SQL.Clear;
    SQL.Add{sSQL):
    Prepare;
    Open:
    if db.Connected then FetchImages;
end;
```

```
procedure TFormT2Images.Edi5KeyDown(Sender: TObject; var Key: Word;
FieldByName('T2SlmageID').AsImageID:=GetNewKey('T2Images');
```
Appendix - A, GUI Source code

function TFormT2Images.GetNewKey(tblName : string) : integer:
begin
sp.UnPrepare;
tsp.Active:=true;
sp.StoredProcName:='GENNEWKEY';
sp.Prepare;
sp.ParamByName(TBLNAME').asstring:=tb!Name;
sp.Prepare;
sp.ExecProc;
result:=sp.parambyName{'KEYVALUE').asinteger;
tsp.Commit;
sp.UnPrepare;
end;

procedure TFormT2Images.qT2SAfterPost(DataSet: TDataSet):
begin
  tT2S.CommitRetaining;
end;

procedure TFormT2Images.qT2SDBeforeInsert(DataSet: TDataSet);
begin
  with qT2S do
  begin
    if not Active then //make sure the master dataset is active
      begin
        ShowMessage('Dataset is not active !');
        abort;
      end;
    if FieldByName('T2SlmageID').IsNull then insert;
    end;
end;

procedure TFormT2Images.qT2SAfterInsert(DataSet: TDataSet);
begin
  with DATASET do
  begin
    FieldByName('FileID').AsInteger:=qT2S.FieldByName('FileID').AsInteger;
    post;
    Close;
    Open;
    locate('FileID',fid, D);
  end;
end;

procedure TFormT2Images.Delete1Click(Sender: TObject);
begin
  with qT2SD do
  begin
    if not Active then //make sure the master dataset is active
      begin
        ShowMessage('Dataset is not active !');
        abort;
      end;
    if FieldByName('FileID').IsNull then
      begin
        ShowMessage('Select an image, then try again !');
        abort;
        end;
      delete;
  end;
end;

procedure TFormT2Images.Cancel1Click(Sender: TObject);
begin
  with qT2SD do
  begin
    ShowMessage('Dataset is not active !');
    abort;
    if FieldByName('FileID').IsNull then
      begin
        ShowMessage('Select an image, then try again !');
        abort;
        end;
      delete;
  end;
end;

procedure TFormT2Images.DBLGrid1ButtonClick(Sender: TObject);
begin
  var fid : integer;
  with qim do
  begin
    if not Active then
      begin
        ShowMessage('Dataset is not active !');
        abort;
        end;
      if FieldByName('FileID').IsNull then
        begin
          ShowMessage('Select an image, then try again !');
          abort;
          end;
        delete;
  end;
end;

procedure TFormT2Images.CancelClick(Sender: TObject);
begin
  with qT2SD do
  begin
procedure TFormT2Slmages.sbGetGDBClick(Sender: TObject);
begin
  With OpenDialog1 do
  begin
    Filter:='Databases|*.GDB|All files|*.*';
    FilterIndex:=1;
    Title:='Select Database File';
    InitialDir:=ExtractFilePath(Application.ExeName);
    if not OpenDialog1.Execute then exit;
    Edit2.Text:=OpenDialog1.FileName;
  end;
end;

procedure TFormT2Slmages.SpeedButton3Click(Sender: TObject);
begin
  ProcessT2sAndMoImages;
end;

procedure TFormT2Slmages.ProcessT2sAndMoImages;
var
  Sigma, X, Y : array of double;
  N, W, H : Integer; //dimension
  a1, a2 : double;
  S, Sx, Sy, Sxx, Sxy, Delta : double;
  Img : array of TImage;
  Mo, T2S : Integer;

#define all functions
function SumSigma(S : array of Double) : Double; //S is Sigma
var k : integer;
begin
  result:=0;
  for k:=0 to N-1 do
    result:=result+(1/(S[k]*S[k]));
end;

function SumSx(X,S : array of Double) : Double;
var k : integer;
begin
  result:=0;
  for k:=0 to N-1 do
    result:=result+(X[k]/(S[k]*S[k]));
end;

function SumSy(Y,S : array of Double) : Double;
var k : integer;
begin
  result:=0;
  for k:=0 to N-1 do
    result:=result+(Y[k]/(S[k]*S[k]));
end;

function SumSxx(X,S : array of Double) : Double;
var k : integer;
begin
  result:=0;
  for k:=0 to N-1 do
    result:=result+(X[k]*X[k]/(S[k]*S[k]));
end;

function SumSxy(X,Y,S : array of Double) : Double;
var k : integer;
begin
  result:=0;
  for k:=0 to N-1 do
    result:=result+(X[k]*Y[k]/(S[k]*S[k]));
end;

//define database checks
function DataChecked : boolean;
begin
  result:=false;
  with QT2SD do
  begin
    //check if dataset is active
    if not Active then
      begin
        ShowMessage('DataSet is not active !');
        exit;
      end;
    //check if any data
    if not Eof then Last;
    N:=RecordCount;
    if N=0 then
      begin
        ShowMessage('No record in DataSet!');
        exit;
      end;
    First;
  end;
  result:=true;
end;

procedure SetDynamicArray;
begin
  SetLength(Sigma,N);
  SetLength(X,N);
  SetLength(Y,N);
  SetLength(Img,N+2);
end;

procedure InitImages;
begin
  for k:=0 to N+1 do
    begin
      end;
end;
img[k]:=Tlmage.Create(nil):
lmg[k].AutoSize:=false;
lmg[k].Stretch:=false;
lmg[k].Parent:=ScrBox1; //image (N) and (N+1) are T2' end Mo end;

procedure LoadDBValues;
var j,k: integer;//jcur: integer;
begin
with qT2SD do
begin
if not bof then First;
k:=0; j:=0;
//lmax:=0;
while not Eof do
begin
W:=rteIdbyname('IMAGEWIDTH').asinteger;
H:=fieldbyname('IMAGEHEIGHT').asinteger;
write blank
if k=0 then
begin
lmg[N].Width:=W;
lmg[N].Height:=H;
lmg[N].Left:=10;
j:=j+5;
lmg[N].Top:=j;
end;
lmg[N+1].Width:=W;
lmg[N+1].Height:=H;
lmg[N+1].Left:=15+W;
lmg[N+1].Top:=j;
j:=j+H;
end;
//if k>0 then lmax:=(imax div k);
Application.ProcessMessages;
end;
end;

procedure CalculateT2SAndMolmages;
type TColorOverlay = record
  case integer of
    1: (IVal: integer):
    2: (b: ARRAY(1..4) OF BYTE);
end;
var oCL : TColorOverlay; k,i,j : integer;
Imax : Integer;
begin
  //progress indicator
  Gaugel .Progress:=0;
  Gaugel .MaxValue:=H;
  Gaugel .ShowText:=true;
  //init sigma
  for k:=0 to N-1 do
  begin
    Sigma[k]:=StrToFloat(eSigma.text);
  end;
  S:=SumSigma(Sigma);
  Sxx:=SumSxx(X,Sigma);
  Sx:=SumSx(X,Sigma);
  lmax:=0;
  for i:=0 to H-1 do
  begin
    for j:=0 to W-1 do
    begin
      T2S:=0; Mo:=0;
      for k:=0 to N-1 do
      begin
        ocl.lVal:=lmg[k].Canvas.Pixels[i,j];
        Y[k]:=ocl.b[2]; //Ln(ocl.b[2]);
        if lmax<ocl.b[2] then lmax:=ocl.b(2]; //identify the maximum intensity
      end;
      Sy:=SumSy(Y,Sigma);
      //if Imax<ocl.b[2] then Imax:=ocl.b[2];
      X[i] :=fieldbyname('ECHOTIME').asinteger;
      Imax:=max[MaxInt,R(k)];
      if Imax<ocl.b[2] then Imax:=ocl.b[2];
      end;
      //end;
      //end;
      Sxy:=SumSxy(X,Y,Sigma):
      Delta:=S*Sxx-Sx*Sx;
      if Delta<0 then
      begin
        a1 :=(Sxx*Sy-Sx*Sxy)/Delta;
        a2:=(S*Sxy-Sx*Sy)/Delta;
      end
      else begin
        a1 :=0; //set to zero or Max?
        a2:=0;
      end;
      if a2=0 then T2S:=0
      else T2S:=trunc(abs(-1/a2));
if $T2S > 255$ then $T2S \rightarrow \text{max int}$ to max intensity

$\text{oCL}[1]:=T2S; \text{oCL}[2]:=T2S; \text{oCL}[3]:=T2S; \text{oCL}[4]:=0;$

$\text{Img}[N].\text{Canvas}.\text{Pixels}[i,j]=\text{oCL}.\text{Val} \quad \//T2S \text{ image}$

if $a1 > 10$ then $a1:=10;

\text{Mo}:=\text{trunc}(|\text{Exp}(a1)|);

if $\text{Mo} > 255$ then $\text{Mo}:=255;

\text{oCL}[1]:=\text{Mo}; \text{oCL}[2]:=\text{Mo}; \text{oCL}[3]:=\text{Mo}; \text{oCL}[4]:=0;$

$\text{Img}[N+1].\text{Canvas}.\text{Pixels}[i,j]=\text{oCL}.\text{Val} \quad \//\text{Mo image}$

procedure SaveT2SAndMImages;
begin
with $\text{qT2S}$ do
begin
Edit;
end;
end;

procedure TformT2Slmages.Menultem1Click(Sender: TObject);
begin
$\text{DBImage3.Center}=\text{Menultem1.Checked};$
$\text{DBImage4.Center}=\text{Menultem1.Checked};$
end;

procedure TformT2Slmages.Menultem2Click(Sender: TObject);
begin
Menultem2.Checked:=not Menultem2,Checked;
$\text{DBImage3.Stretch}=\text{Menultem2.Checked};$
$\text{DBImage4.Stretch}=\text{Menultem2.Checked};$
end;

procedure TformT2Slmages.Menultem3Click(Sender: TObject);
begin
Menultem3.Checked:=not Menultem3,Checked;
$\text{DBImage2.Center}=\text{Menultem3.Checked};$
end;

procedure TformT2Slmages.Menultem4Click(Sender: TObject);
begin
Menultem4.Checked:=not Menultem4,Checked;
$\text{DBImage2.Stretch}=\text{Menultem4.Checked};$
end;

procedure TformT2Slmages.Button1Click(Sender: TObject);
begin
LoadFields(cbField1);
end;

procedure TformT2Slmages.LoadFields(cbFields: TComboBox);
var i, cnt : Integer;
begin
for $k:=0$ to $N+1$ do
begin
$\text{Img}[k].\text{Free};$
end;
end;

procedure TformT2Slmages.FreeImages;
var k : integer;
begin
for $k:=0$ to $N+1$ do
begin
$\text{Img}[k].\text{Free};$
end;
end;

procedure TformT2Slmages.SetDynamicArray;
var $\text{FieldAdd}$:
begin
$\text{SetDynamicArray}();$
end;

procedure TformT2Slmages.Initlmages;
var $\text{FieldData};$
begin
$\text{Initlmages}();$
end;

procedure TformT2Slmages.LoadDBValues;
begin
$\text{LoadDBValues}();$
end;

procedure TformT2Slmages.CalculateT2SAndMImages;
begin
$\text{CalculateT2SAndMImages}();$
end;

procedure TformT2Slmages.SaveT2SAndMImages;
begin
$\text{SaveT2SAndMImages}();$
end;

end;
begin
with qlm do
begin
if not Active then
begin
ShowMessage('DataSet is not active !');
exit;
end;
end;
cnt:=FieldDefs.Count;
//cbFT.Clear;
for i:=0 to cnt-1 do
begin
if not (FieldDefs[i].DataType in [ftBlob, ftMemo, ftGraphic, ftFmtMemo, ftParadoxOle, ftDbaseOle]) then
begin
cbFields.Items.Add(Fields[i].FieldName);
end;
end;
end;
end;

procedure TFormT2Images.dsT2SStateChange(Sender: TObject);
begin
if dsT2S.State in [dsEdit, dslnsert] then
begin
DBLookupComboBox1.Enabled:=false;
end
else begin
DBLookupComboBox1.Enabled:=true;
end;
end;
end.
Visual GUI module

object formT2Images: TFormT2Images
Left = 45
Top = 46
Width = 908
Height = 626
Caption = 'T2* Images (Plug-Ins)'
Color = clBtnFace
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = [fsBold]
FormStyle = fsMDIChild
OldCreateOrder = False
Position = poDefault
Visible = True
WindowState = wsMaximized
OnClose = FormClose
OnCreate = FormCreate
PixelsPerlnch = 96
TextHeight = 13

object PageControl1: TPageControl
Left = 0
Top = 41
Width = 900
Height = 558
ActivePage = TabSheet1
Align = alClient
TabOrder = 0

object TabSheet1: TTabSheet
Caption = 'Select Images'
object Splitter2: TSplitter
Left = 625
Top = 273
Height = 257
end

object Panel2: TPanel
Left = 0
Top = 0
Width = 892
Height = 273
Align = alTop
BevelOuter = bvNone
BorderWidth = 2
TabOrder = 0

object GroupBox1: TGroupBox
Left = 2
Top = 2
Width = 479
Height = 269
Align = alLeft
Caption = 'Open Database'
TabOrder = 0

object Label2: TLabel
Left = 24
Top = 32
Width = 34
Height = 13
Caption = 'Server: '

object Label1: TLabel
Left = 24
Top = 56
Width = 49
Height = 13
Caption = 'Database: '

object sbGetGDB: TSpeedButton
Left = 400
Top = 56
Width = 23
Height = 22
Hint = 'Get database file.'
Caption = '
ParentShowHint = False
ShowHint = True
OnClick = sbGetGDBClick

object Labels: TLabel
Left = 24
Top = 80
Width = 36
Height = 13
Caption = 'User id: '

object Label4: TLabel
Left = 24
Top = 104
Width = 49
Height = 13
Caption = 'Password: '

object sbSaveCSD: TSpeedButton
Left = 80
Top = 128
Width = 49
Height = 22
Hint = 'Save database server login details.'
Caption = 'Save'
ParentShowHint = False
ShowHint = True
OnClick = sbSaveCSDClick

object sbLoadCSD: TSpeedButton
Left = 128
Top = 128
Width = 49
Height = 22
Hint = 'Load database server login details.'
Caption = 'Load'
ParentShowHint = False
ShowHint = True
object sbTestCDSD: TSpeedButton
Left = 184
Top = 128
Width = 41
Height = 22
Hint = 'Open database connection.'
Caption = 'Open'
ParentShowHint = False
ShowHint = True
OnClick = sbTestCDSDClick
end

object Labels: TLabel
Left = 80
Top = 160
Width = 48
Height = 13
Caption = 'File name:'
Visible = False
end

object SpeedButtonl: TSpeedButton
Left = 366
Top = 232
Width = 51
Height = 22
Hint = 'Search images'
Caption = 'Search'
ParentShowHint = False
ShowHint = True
OnClick = SpeedButtonlClick
end

object Label?; TLabel
Left = 144
Top = 160
Width = 41
Height = 13
Caption = 'Seq. file:'
Visible = False
end

object SpeedButton2; TSpeedButton
Left = 416
Top = 232
Width = 49
Height = 22
Hint = 'Search images'
Caption = 'Clear'
ParentShowHint = False
ShowHint = True
OnClick = SpeedButton2Click
end

object Labels: TLabel
Left = 272
Top = 212
Width = 23
Height = 13
Caption = 'AND'
end

object Label10: TLabel
Left = 192
Top = 160
Width = 60
Height = 13
Caption = 'Echo time:'
Visible = False
end

object Label12: TLabel
Left = 8
Top = 160
Width = 41
Height = 13
Caption = 'Search'
Font.Charset = DEFAULT_CHARSET
Font.Color = clNavy
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = [fsBold]
ParentFont = False
end

object Edit2: TEdit
Left = 80
Top = 56
Width = 313
Height = 21
BorderStyle = bsNone
TabOrder = 0
end

object Edit1: TEdit
Left = 80
Top = 32
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 1
end

object Edits: TEdit
Left = 80
Top = 80
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 2
end

object Edit4: TEdit
Left = 80
Top = 104
Width = 145
Height = 21
BorderStyle = bsNone
TabOrder = 3
end

object Label9: TLabel
Left = 272
Top = 212
Width = 23
Height = 13
Caption = 'AND'
end
PasswordChar = '$'
TabOrder = 3
end

object RadioButton1: TRadioButton
Left = 240
Top = 32
Width = 57
Height = 17
Caption = 'Local'
Checked = True
TabOrder = 4
TabStop = True
end

object RadioButton2: TRadioButton
Left = 304
Top = 32
Width = 73
Height = 17
Caption = 'Remote'
TabOrder = 5
end

object Edits: TEdit
Left = 184
Top = 184
Width = 81
Height = 21
BorderStyle = bsNone
Color = 14416381
TabOrder = 6
OnKeyDown = EditSKeyDown
end

object EditE: TEdit
Left = 184
Top = 208
Width = 81
Height = 21
BorderStyle = bsNone
Color = 14416381
TabOrder = 7
OnKeyDown = EditSKeyDown
end

object cbFNI: TComboBox
Left = 120
Top = 184
Width = 57
Height = 21
BevelInner = bvNone
BevelKind = bkFlat
BevelOuter = bvNone
Color = 14416381
ItemHeight = 13
TabOrder = 8
Items.Strings = ('Like'
'&lt;'
'&gt;')
end

object cbSFI: TComboBox
Left = 120
Top = 208
Width = 57
Height = 21
BevelInner = bvNone
BevelKind = bkFlat
BevelOuter = bvNone
Color = 14416381
ItemHeight = 13
TabOrder = 9
Items.Strings = ('Like'
'&lt;'
'&gt;')
end

object cbFN2: TComboBox
Left = 304
Top = 184
Width = 57
Height = 21
BevelInner = bvNone
BevelKind = bkFlat
BevelOuter = bvNone
Color = 14416381
ItemHeight = 13
TabOrder = 10
Items.Strings = ('Like'
'&lt;'
'&gt;')
end

object cbSF2: TComboBox
Left = 304
Top = 208
Width = 57
Height = 21
BevelInner = bvNone
BevelKind = bkFlat
BevelOuter = bvNone
Color = 14416381
ItemHeight = 13
TabOrder = 11
Items.Strings = ('Like'
'&lt;'
'&gt;')
end

object Edit7: TEdit
Left = 366
Top = 184
Width = 99
Height = 21
BorderStyle = bsNone
Color = 14416381
TabOrder = 12
OnKeyDown = EditSKeyDown
end
object cbFL1: TComboBox
  Left = 8
  Top = 184
  Width = 105
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 18
  Items.Strings = (
    'Like'
    '<'
    '->'
  )
end

eobject cbFL2: TComboBox
  Left = 8
  Top = 208
  Width = 105
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 19
  Items.Strings = (
    'Like'
    '<'
    '->'
  )
end

eobject cbFL3: TComboBox
  Left = 8
  Top = 232
  Width = 105
  Height = 21
  BevelInner = bvNone
  BevelKind = bkFlat
  BevelOuter = bvNone
  Color = 14416381
  ItemHeight = 13
  TabOrder = 20
  Items.Strings = (
    'Like'
    '<'
    '->'
  )
end

object Button1: TButton
  Left = 344
  Top = 120
  Width = 121
  Height = 25
  Caption = 'Test - Load Fields'
  TabOrder = 17
  Visible = False
  OnClick = Button1Click
end

object GroupBox2: TGroupBox
  Left = 481
  Top = 2
  Width = 409
  Height = 269
Align = alClient
Caption = 'Selected Images'
TabOrder = 1

object Label1: TLabel
Left = 8
Top = 24
Width = 91
Height = 13
Caption = 'Image group name:
FocusControl = DBEdit
Font.Charset = DEFAULT_CHARSET
Font.Color = cIMaroon
Font.Height = -11
Font.Name = 'MS Sans Serif
Font.Style = [ ]
ParentFont = False
dn
end

object DBLookupComboBox1: TDBLookupComboBox
Left = 104
Top = 24
Width = 153
Height = 21
BevelOuter = bvNone
KeyField = 'T2SIMAGETITLE'
ListField = 'T2SIMAGETITLE'
ListSource = dsT2S
TabOrder = 3
dn
end

object DBEdit1: TDBEdit
Left = 104
Top = 24
Width = 137
Height = 21
BorderStyle = bsNone
DataField = 'T2SIMAGETITLE'
DataSource = dsT2S
TabOrder = 0
end

object DBNavigator1: TDBNavigator
Left = 264
Top = 22
Width = 140
Height = 26
DataSource = dsT2S
VisibleButtons = [nbPrior, nbNext, nbInsert, nbDelete, nbEdit, nbPost, nbCancel]
Flat = True
TabOrder = 1
dn
end

object DBGrid2: TDBGrid
Left = 2
Top = 56
Width = 405
Height = 211
Align = alBottom
BorderStyle = bsNone
DataSource = dsT2SD
Options = [dgTitles, dgIndicator, dgColumnResize, dgColLines, dgRowLines, dgTabs, dgRowSelect, dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
PopupMenu = pmT2S
ReadOnly = True
TabOrder = 2
TitleFont.Charset = DEFAULT_CHARSET
TitleFont.Color = cWindowText
TitleFont.Height = -11
TitleFont.Name = 'MS Sans Serif
TitleFont.Style = [ ]
OnTitleClick = DBGrid1TitleClick
Columns = <
Item
Expanded = False
FieldName = 'MAXINTENSITY'
Title.Caption = 'Max Intensity'
Width = 60
Visible = True
dn
end
Item
Expanded = False
FieldName = 'FILENAME'
Title.Caption = 'File Name'
Width = 60
Visible = True
dn
end
Item
Expanded = False
FieldName = 'ECHOTIME'
Title.Caption = 'E Time'
Width = 40
Visible = True
dn
end
Item
Expanded = False
FieldName = 'FILESEQUENCE'
Title.Caption = 'File Sequence'
Width = 266
Visible = True
dn
end
Item
Expanded = False
FieldName = 'FILEFULLNAME'
Title.Caption = 'File Full Name'
Width = 80
Visible = True
dn
end
Item
Expanded = False
FieldName = 'IMAGEWIDTH'
Title.Caption = 'Width'
Width = 40
Visible = True
dn
end
Item
Expanded = False
FieldName = 'IMAGEHEIGHT'
Title.Caption = 'Height'
Width = 40
Visible = True
dn
end

end

end
Title.Caption = 'Height'
Width = 42
Visible = True
end
item
Expanded = False
FieldName = 'FILEID'
Title.Caption = 'Image ID'
Width = 46
Visible = True
end>
end

object DBGridI: TDBGrid
Left = 0
Top = 273
Width = 625
Height = 257
Align = alLeft
BorderStyle = bsNone
DataSource = dsim
Options = [dgTitles, dgIndicator, dgColumnResize,
dgColLines, dgRowLines, dgTabs, dgRowSelect,
dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
ReadOnly = True
TabOrder = 1
TitleFont.Charset = DEFAULT_CHARSET
TitleFont.Color = clWindowText
TitleFont.Height = -11
TitleFont.Name = 'MS Sans Serif'
TitleFont.Style = [qBold]
OnDblClick = DBGridIDblClick
OnTitleClick = DBGridITitleClick
Columns = <
item
Expanded = False
FieldName = 'MAXINTENSITY'
Title.Caption = 'Max Intensity'
Width = 69
Visible = True
end
item
Expanded = False
FieldName = 'FILENAME'
Title.Caption = 'File Name'
Width = 90
Visible = True
end
item
Expanded = False
FieldName = 'ECHOTIME'
Title.Caption = 'E Time'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'FILESEQUENCE'
Title.Caption = 'File Sequence'
Width = 248
Visible = True
end
item
Expanded = False
FieldName = 'FILEFULLNAME'
Title.Caption = 'File Full Name'
Width = 80
Visible = True
end
item
Expanded = False
FieldName = 'IMAGEWIDTH'
Title.Caption = 'Width'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'IMAGEHEIGHT'
Title.Caption = 'Height'
Width = 42
Visible = True
end>
end

object PageControl2: TPageControl
Left = 628
Top = 273
Width = 264
Height = 257
ActivePage = TabSheetS
Align = alClient
TabOrder = 2
object TabSheetS: TTabSheet
Caption = 'Image'
object DBImagel: TDBImage
Left = 0
Top = 0
Width = 256
Height = 229
Align = alClient
Color = cIBIack
DataField = 'IMAGE'
DataSource = dsIm
PopupMenu = PopupMenul
TabOrder = 0
end
end
end

object TabSheet3: TTabSheet
Caption = 'Image'
object DBImage1: TDBImage
Left = 0
Top = 0
Width = 256
Height = 229
Align = alClient
Color = cIBIack
DataField = 'IMAGE'
DataSource = dsIm
PopupMenu = PopupMenu1
TabOrder = 0
end
end
end

object TabSheet4: TTabSheet
T2* plug-in module

Caption = 'Header Info'
imagindex = 1
object DBMemo1: TDBMemo
Left = 0
Top = 0
Width = 261
Height = 229
Align = alClient
DataField = 'FILEHEADER'
DataSource = dsim
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Courier'
Font.Style = [q]
ParentFont = False
ScrollBars = ssBoth
TabOrder = 0
end
end
end
object TafaSheet2: TTabSheet
Caption = 'Process T2* Image'
Imagindex = 1
object Splitter1: TSplitter
Left = 625
Top = 265
Height = 265
end
object Panels: TPanel
Left = 0
Top = 0
Width = 892
Height = 265
Align = alTop
BevelOuter = bvLowered
TabOrder = 0
end
object SpeedButtonS: TSpeedButton
Left = 584
Top = 5
Width = 153
Height = 22
Hint = 'Calculate T2* and Mo Images'
Caption = 'Calculate T2* and Mo Images'
ParentShowHint = False
ShowHint = True
OnClick = SpeedButtonSCIick
end
object Label13: TLabel
Left = 760
Top = 8
Width = 32
Height = 13
Caption = 'Sigma:'
end
object Gaugel: TGauge
Left = 872
Top = 32
Width = 17
Height = 217
BackColor = clBtnFace
BorderStyle = bsNone
ForeColor = clMaroon
Kind = gkVerticalBar
Progress = 0
ShowText = False
end
object DBGrid3: TDBGrid
Left = 1
Top = 8
Width = 208
Height = 249
DataSource = dsT2S
Options = [dgTitles, dgColumnResize, dgColLines, dgFlowLines, dgAutoSize, dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
ReadOnly = True
TabOrder = 0
TitleFont.Charset = DEFAULT_CHARSET
TitleFont.Color = clWindowText
TitleFont.Height = -11
TitleFont.Name = 'MS Sans Serif'
TitleFont.Style = [q]
OnTitleClick = DBGridlTitleClick
Columns = <
item
Expanded = False
FieldName = 'T2SIMAGETITLE'
Caption = 'Image Group Name'
Width = 136
Visible = True
end
item
Expanded = False
FieldName = 'T2SIMAGEID'
Caption = 'Group ID'
Width = 54
Visible = True
end>
end
object PageControl4: TPageControl
Left = 224
Top = 8
Width = 545
Height = 240
ActivePage = TabSheet7
Style = tsButtons
TabOrder = 1
end
object TabSheet7: TTabSheet
Caption = 'T2* Image'
object DBImageS: TDBImage
Left = 0
Top = 0
Width = 337
Height = 217
Align = alClient
Color = cBlack
DataField = 'T2SIMAGE'
DataSource = dsT2S
PopupMenu = PopupMenu2
TabOrder = 0
end
end
object TabSheetS: TTabSheet
Caption = 'Mo Image'
ImageIndex = 1
object DBImage4: TDBImage
Left = 0
Top = 0
Width = 337
Height = 217
Align = alClient
Color = cBlack
DataField = 'MOIMAGE'
DataSource = dsT2S
PopupMenu = PopupMenu2
TabOrder = 0
end
end
object ScrollBoxI: TScrollBox
Left = 584
Top = 32
Width = 281
Height = 217
TabOrder = 2
end
object eSigma: TEdit
Left = 800
Top = 8
Width = 65
Height = 21
TabOrder = 3
Text = '1'
end
object DBGrid4: TDBGrid
Left = 0
Top = 265
Width = 625
Height = 265
Align = alLeft
BorderStyle = bsNone
DataSource = dsT2SD
Options = [dgTitles, dgColumnResize, dgColLines, dgRowLines, dgTabs, dgRowSelect, dgAlwaysShowSelection, dgConfirmDelete, dgCancelOnExit]
ReadOnly = True
TabOrder = 1
TitleFont.Charset = DEFAULT_CHARSET
TitleFont.Color = clWindowText
TitleFont.Height = 11
TitleFont.Name = 'MS Sans Serif'
TitleFont.Style = []
OnTitleClick = DBGrid4TitleClick
Colors = <
item
Expanded = False
FieldName = 'MAXINTENSITY'
Title.Caption = 'Max Intensity'
Width = 69
Visible = True
end
item
Expanded = False
FieldName = 'FILENAME'
Title.Caption = 'File Name'
Width = 80
Visible = True
end
item
Expanded = False
FieldName = 'ECHOTIME'
Title.Caption = 'E Time'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'FILESEQUENCE'
Title.Caption = 'File Sequence'
Width = 269
Visible = True
end
item
Expanded = False
FieldName = 'FILEFULLNAME'
Title.Caption = 'File Full Name'
Width = 80
Visible = True
end
item
Expanded = False
FieldName = 'IMAGEWIDTH'
Title.Caption = 'Width'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'IMAGEHEIGHT'
Title.Caption = 'Height'
Width = 40
Visible = True
end
item
Expanded = False
FieldName = 'FILEID'
Title.Caption = 'File ID'
Width = 80
Visible = True
end
end
T2* plug-in module

Title.Caption = 'Image ID'
Width = 48
Visible = True
end

object PageControl1: TPageControl
Left = 828
Top = 265
Width = 264
Height = 286
ActivePage = TabSheet5
Align = alClient
TabOrder = 2
object TabSheet5: TTabSheet
Caption = 'Image'
object DBImage2: TDBImage
Left = 0
Top = 0
Width = 256
Height = 237
Align = alClient
Color = cIBIack
DataField = 'IMAGE'
DataSource = dsT2SD
PopupMenu = PopupMenuS
TabOrder = 0
end
end

object TabSheet6: TTabSheet
Caption = 'Header Info'
Imagelndex = 1
object DBMemo2: TDBMemo
Left = 0
Top = 0
Width = 261
Height = 170
Align = alClient
Color = cIBIack
DataField = 'FILEHEADER'
DataSource = dsT2SD
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'Courier'
Font.Style = q
ParentFont = False
ScrollBars = ssBoth
TabOrder = 0
end

object Panel1: TPanel
Left = 0
Top = 0
Width = 900
Height = 41
Align = alTop
BevelInner = bvRaised
BevelOuter = bvNone
TabOrder = 1
object Label6: TLabel
Left = 232
Top = 13
Width = 132
Height = 13
Caption = 'Not connected to Database'
Font.Color = cINavy
Font.Height = 11
Font.Name = 'MS Sans Serif'
Font.Style = q
ParentFont = False
end

object db: TIBDatabase
Connected = True
DatabaseName = 'E:\Data\HB\UniS\Dev\lmageProc\Data\lmageDB.GDB'
Params.Strings = ('user_name=Sysdba'
       'password=masterkey')
LoginPrompt = False
IdleTimer = 0
SQLDriver = 3
TruncateFiles = False
AllowStreamedConnected = False
Left = 288
Top = 192
end

object tDB; TIBTransaction
Active = False
DefaultDatabase = db
AutoStopAction = saNone
Left = 288
Top = 192
end

object SaveDialog1: TSaveDialog
DefaultExt = '*.d!d'
Filter = 'Database login details|*.d!d|AII files|*.*'
Title = 'Save Database Login Details'
Left = 240
Top = 200
end

object OpenDialog1: TOpenDialog
Filter = 'Databases|*.GDB|AII files|*.*'
Left = 272
Top = 200
end

object tim: TIBTransaction
Active = False
DefaultDatabase = db
AutoStopAction = saNone
Left = 340
Top = 193
end
object qlm: TIBQuery
Database = db
Transaction = tlm
BufferChunks = 1000
CachedUpdates = False
ParamCheck = False
SQL.Strings = (  
'Select I.'  
'From ImageFiles I'  
'Where I.FileID<0'  
')
Left = 372
Top = 393
end

object dsim: TDataSource
DataSet = qlm
Left = 404
Top = 393
end

object Centrel: TMenultem
Caption = 'Centre'
Checked = True
OnClick = Centrel_Click
end

object Stretchl: TMenultem
Caption = 'Stretch'
OnClick = Stretchl_Click
end

object tT2SD: TIBTransaction
Active = True
DefaultDatabase = db
AutoStopAction = saNone
Left = 380
Top = 425
end

object qT2SD: TIBQuery
Database = db
Transaction = tT2SD
AfterDelete = qT2SDAfterPost
AfterInsert = qT2SDAfterlnsert
BeforeInsert = qT2SDBeforelnsert
BufferChunks = 1000
CachedUpdates = False
DataSource = dsT2SD
SQL.Strings = (  
'Select T.'  
'From T2SIMAGEST'  
'Order by T.T2SIMAGETITLE')
UpdateObject = uT2SD
Left = 276
Top = 425
end

object qT2ST2SIMAGElD: TIntegerField
FieldName = 'T2SIMAGEID'
Origin = T2SIMAGES.T2SIMAGEID'
Required = True
end

object qT2ST2SIMAGE: TIBStringField
FieldName = 'T2SIMAGE'
Origin = T2SIMAGES.T2SIMAGE
Size = 8
end

object qT2SM0IMAGE: TIBStringField
FieldName = 'M0IMAGE'
Origin = T2SIMAGES.M0IMAGE
Size = 8
end

object qT2ST2SIMAGETITLE: TIBStringField
FieldName = 'T2SIMAGETITLE'
Origin = T2SIMAGES.T2SIMAGETITLE
Required = True
Size = 50
end

end

object dsT2S: TDataSource
DataSet = qT2S
OnStateChange = dsT2SStateChange
Left = 308
Top = 426
end

object uT2SD: TIBUpdateSQL
RefreshSQL.Strings = (
  Select 
  T2SIMAGEID, 
  FILEID 
  from T2SIMAGESOURCES 
  where
  T2SIMAGEID = :T2SIMAGEID and
  FILEID = :FILEID)
ModifySQL.Strings = (
  update T2SIMAGESOURCES 
  set
  T2SIMAGEID = :T2SIMAGEID, 
  FILEID = :FILEID 
  where
  T2SIMAGEID = :OLD_T2SIMAGEID and
  FILEID = :OLD_FILEID)
InsertSQL.Strings = (
  Insert into T2SIMAGESOURCES 
  (T2SIMAGEID, FILEID) 
  values
  (:T2SIMAGEID, :FILEID))
DeleteSQL.Strings = (
  delete from T2SIMAGESOURCES 
  where
  T2SIMAGEID = :OLD_T2SIMAGEID and
  FILEID = :OLD_FILEID)
Left = 472
Top = 424
end

object uT2S: TIBUpdateSQL
RefreshSQL.Strings = (
  Select 
  T2SIMAGEID, 
  T2SIMAGE, 
  MOIMAGE, 
  T2SIMAGETITLE 
  from T2SIMAGES 
  where
  T2SIMAGEID = :T2SIMAGEID)
ModifySQL.Strings = (
  update T2SIMAGES 
  set
  T2SIMAGEID = :T2SIMAGEID, 
  T2SIMAGE = :T2SIMAGE, 
  MOIMAGE = :MOIMAGE,
  T2SIMAGETITLE = :T2SIMAGETITLE 
  where
  T2SIMAGEID = :OLD_T2SIMAGEID)
InsertSQL.Strings = (
  Insert into T2SIMAGES 
  (T2SIMAGEID, T2SIMAGE, MOIMAGE, T2SIMAGETITLE) 
  values
  (:T2SIMAGEID, :T2SIMAGE, :MOIMAGE, :T2SIMAGETITLE))
DeleteSQL.Strings = (
  delete from T2SIMAGES 
  where
  T2SIMAGEID = :OLD_T2SIMAGEID)
Left = 336
Top = 425
end

object ISP: TIBTransaction
Active = False
DefaultDatabase = db
AutoStopAction = saNone
Left = 532
Top = 302
end

object SP: TIBStoredProc
Database = db
Transaction = tSP
Left = 532
Top = 425
end

object pmT2S: TPopupMenu
Left = 706
Top = 220
end

object InsertItem: TMenuitem
Caption = 'Insert'
OnClick = DBGridIDblClick
end

object N1: TMenultem
Caption = '-'
end

object DeleteItem: TMenultem
Caption = 'Delete'
OnClick = DeleteItemClick
end

object N2: TMenultem
Caption = '-'
end

object CancelItem: TMenultem
Caption = 'Cancel'
OnClick = CancelItemClick
end

object PopupMenu2: TPopupMenu
Left = 442
Top = 156
end

object Menultem1: TMenuitem 
Caption = 'Centre'
Checked = True
OnClick = Menultem1Click
end

object Menultem2: TMenuitem 
Caption = 'Stretch'
end

object N4: TMenultem
```
Caption = 'Clear'
OnClick = N3Click
end

object N3: TMenuItem
Caption = 'Clear'
OnClick = N3Click
end

object PopupMenu3: TPopupMenu
Left = 730
Top = 396
object Menultem3: TMenuItem
Caption = 'Centre'
Checked = True
OnClick = Menultem3Click
end

object Menultem4: TMenuItem
Caption = 'Stretch'
OnClick = Menultem4Click
end
```
Image processing functions

unit unit_SiemensRawImageFn;

interface

uses
    Classes, SysUtils;

Const
    ImageStartPos=6144;

Type
    TValueType=(vtSI, vtLI, vtFS, vtFD, vtS); {small Int, Long Int, FloatDouble, FloatSingle, String}

function GetAbsBufferPos(BlockNum : SmallInt; RelativePos : Integer): Integer;

function GetHeaderData(Stream : TFileStream; vt : TValueType; BlockNum : SmallInt; sPos, ePos : Integer) : Variant;

procedure ImageHeaderDetails(Stream : TFileStream; HeaderDetail : TStringList);

implementation

function GetAbsBufferPos(BlockNum : SmallInt; RelativePos : Integer) : Integer;
var
    dBlock : array[1..6] of Integer;
begin
    dBIOck[1]:= -1; dBIOck[2]:= 1023; dBIOck[3]:= 2047;
    dBIOck[4]:= 3199; dBIOck[5]:= 4223; dBIOck[6]:= 5247;
    result:= dBIOck[BlockNum]+RelativePos;
end;

function GetHeaderData(Stream : TFileStream; vt : TValueType; BlockNum : SmallInt; sPos, ePos : Integer) : Variant;
var
    SI : SmallInt; I, TotalBuffer : Integer; fS : Single; d : Double; s : String;
begin
    TotalBuffer:= ePos-sPos+1;
    Stream.Position:= GetAbsBufferPos(BlockNum, sPos);
    case vt of
    vtSI:
        begin
            Stream.ReadBuffer(SI, TotalBuffer);
            result:= SI;
            exit;
        end;
    vtLI:
        begin
            Stream.ReadBuffer(I, TotalBuffer);
        end;
    vtFS:
        begin
            Stream.ReadBuffer(fS, TotalBuffer);
        end;
    vtFD:
        begin
            Stream.ReadBuffer(d, TotalBuffer);
        end;
    end;
end;

procedure Swap_Endian; //Delphi does not have this function, I had to write one

var
    X : Integer;
    dv1, dv2 : TDoubleOverlay;
    sv1, sv2 : TSingleOverlay;
    lv1, lv2 : TLongIntOverlay;
begin
    case vt of
    vtSI:
        begin
            SI:= swap(SI);
        end;
    vtLI:
        begin
            lv1.LVal:= l;
            for x:=1 to 4 do lv2.b[x]:= lv1.b[5-x];
            l:= lv2.LVal;
        end;
    vtFS:
        begin
            sv1.SVal:= fS;
            for x:=1 to 4 do sv2.b[x]:= sv1.b[5-x];
            fS:= sv2.SVal;
        end;
    vtFD:
        begin
            dv1.dVal:= d;
            for x:=1 to 8 do dv2.b[x]:= dv1.b[9-xl;
            d:= dv2.dVal;
        end;
    end;
end;
Swap_Endian;
result:=l;
end;

VtFS:
begin
  fStream.ReadBuffer(fS, TotalBuffer);
  Swap_Endian;
  result:=fS;
  exit;
end;

VtFD:
begin
  fStream.ReadBuffer(d, TotalBuffer);
  Swap_Endian;
  result:=d;
  exit;
end;
VtS:
begin
  SetLength(s, TotalBuffer);
  for l:=1 to TotalBuffer do
    begin
      fStream.ReadBuffer(s[l],1);
      if s[l]=#0 then s[l]:=' ';
    end;
s:=trim(s);
  result:=s;
  exit;
end;
end;

end;

procedure lmageHeaderDetails(fs :TFileStream;
HeaderDetail: TStringList);
var Yr, Mn, Dy, Hr, Mi, Se, Ms : word;
procedure WriteToDisplay(s : string);
begin
  HeaderDetail.Add(trim(s));
end;
begin
  //HeaderDetail:**;
  try
    //read block 1
    WriteToDisplay('{ Block 1}');
    Yr:=GetHeaderData(fs, vtLI, 1, 1, 4);
    Mn:=GetHeaderData(fs, vtLI, 1, 5, 8);
    Dy:=GetHeaderData(fs, vtLI, 1, 9, 12);
    WriteToDisplay('Study date, year : '+IntToStr(Yr));
    WriteToDisplay('Study date, month : '+IntToStr(Mn));
    WriteToDisplay('Study date, day : '+IntToStr(Dy));
  except
    on e: Exception do
      WriteToDisplay('Error: '+e.Message);
  end;

  WriteToDisplay('Study time, hour : '+IntToStr(Hr));
  WriteToDisplay('Study time, minute : '+IntToStr(Mi));
  WriteToDisplay('Study time, second : '+IntToStr(Se));
  WriteToDisplay('Study time, mil. sec. : '+IntToStr(Ms));
end;

WriteToDisplay('Study time :
*FormatDateTime(Chh:mm:ss', EncodeTime(Hr,Mi,Se,Ms)));

WriteToDisplay('Acquis. time, hour : '+IntToStr(Hr));
WriteToDisplay('Acquis. time, minute : '+IntToStr(Mi));
WriteToDisplay('Acquis. time, second : '+IntToStr(Se));
WriteToDisplay('Acquis. time, mil. sec. : '+IntToStr(Ms));
end;

WriteToDisplay('Acquis. time :
*FormatDateTime(Chh:mm:ss', EncodeTime(Hr,Mi,Se,Ms)));

WriteToDisplay('Image date, year : '+IntToStr(Yr));
WriteToDisplay('Image date, month : '+IntToStr(Mn));
WriteToDisplay('Image date, day : '+IntToStr(Dy));
WriteToDisplay('Image date :
*FormatDateTime(Chh:mm:ss', EncodeTime(Hr,Mi,Se,Ms)));

WriteToDisplay('Data Set Subtype(1) :
*IntToStr(GetHeaderData(fs, vtLI, 1, 85, 88)));
WriteToDisplay('Data Set Subtype(2) :
*IntToStr(GetHeaderData(fs, vtLI, 1, 89, 92)));
WriteToDisplay('Modality :
*IntToStr(GetHeaderData(fs, vtLI, 1, 93, 96)));
WriteToDisplay('Scanner Manufacturer :
*GetHeaderData(fs, vtS, 1, 97, 105));
WriteToDisplay('Hospital :
*GetHeaderData(fs, vtS, 1, 106, 132));
WriteToDisplay('Referring Physician :
*GetHeaderData(fs, vtS, 1, 133, 159));
WriteToDisplay('Station ident. :
*GetHeaderData(fs, vtS, 1, 160, 186));
WriteToDisplay('Procedure Descrip. (1) :
*GetHeaderData(fs, vtS, 1, 187, 213));
WriteToDisplay('Procedure Descrip. (2) :
*GetHeaderData(fs, vtS, 1, 214, 240));
WriteToDisplay("Admitting Diagnosis:");
WriteToDisplay("Manufacturer Model:");
WriteToDisplay("Manufacturer:");
WriteToDisplay("Scan Sequence:");
WriteToDisplay("Contrast:");
WriteToDisplay("Slice Thickness (mm):");
WriteToDisplay("Acquisition Columns:");
WriteToDisplay("Number of Averages:");
WriteToDisplay("Reconstruction Time (ms):");
WriteToDisplay("Number of Prescans:");
WriteToDisplay("Tot. Satur. Regions:");
WriteToDisplay("Field of View Ratio:");
WriteToDisplay("Energy Dose Limit:");
WriteToDisplay("Calculation Submode:");
WriteToDisplay("Field of View Ratio:");
WriteToDisplay("ADC Offset Real Part:");
WriteToDisplay("ADC Offset Imag. Part:");
WriteToDisplay("Receive Total Gain (dB5):");
WriteToDisplay("Receive PreA. Gain (dB5):");
WriteToDisplay("Receive Ampl. Gain (dB5):");
WriteToDisplay("Receive Total Gain (dB5):");
WriteToDisplay("Receive PreA. Gain (dB5):");
WriteToDisplay("Receive Ampl. Gain (dB5):");
WriteToDisplay("Receive Total Gain (dB5):");
WriteToDisplay("Receive PreA. Gain (dB5):");
WriteToDisplay("Receive Ampl. Gain (dB5):");
WriteToDisplay("Receive Total Gain (dB5):");
WriteToDisplay("Receive PreA. Gain (dB5):");
WriteToDisplay("Receive Ampl. Gain (dB5):");
WriteToDisplay("Receive Total Gain (dB5):");
WriteToDisplay("Receive PreA. Gain (dB5):");
WriteToDisplay("Receive Ampl. Gain (dB5):");
WriteToDisplay("Receive Total Gain (dB5):");
WriteToDisplay("Receive PreA. Gain (dB5):");
WriteToDisplay("Receive Ampl. Gain (dB5):");
WriteToDisplay("3D Phase Csmpl. Part : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 825, 828)));
WriteToDisplay("3D Phase Csmpl. Lin. : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 821, 824)));
WriteToDisplay("3D Phase Csmpl. Part : "+
IntToStr(GetHeaderData(fs, vtSI, 4, 821, 824)));
WriteToDisplay("3D Phase Csmpl. Lin. : "+
IntToStr(GetHeaderData(fs, vtSI, 4, 817, 820)));
WriteToDisplay("3D Phase Csmpl. Part : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 813, 816)));
WriteToDisplay("3D Phase Csmpl. Lin. : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 809, 812)));
WriteToDisplay("3D Phase Csmpl. Part : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 805, 808)));
WriteToDisplay("3D Phase Csmpl. Lin. : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 801, 804)));
WriteToDisplay("3D Phase Csmpl. Part : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 793, 796)));
WriteToDisplay("3D Phase Csmpl. Lin. : "+
IntToStr(GetHeaderData(fs, vtLI, 4, 789, 792)));
WriteToDisplay('Gate Threshold (%) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 835, 844)));
WriteToDisplay('Gate Ratio (%) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 897, 906)));
WriteToDisplay('Total Interp. Images : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 905, 908)));
WriteToDisplay('Total Echos : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 909, 912)));
WriteToDisplay('Second Echo Time (ms) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 916, 923)));
WriteToDisplay('Second Rep. Time (ms) : ' +
FloatToStr(GetHeaderData(fs, vtFD, 4, 924, 931)));
WriteToDisplay('Cardiac Code : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 937, 940)));
WriteToDisplay('Order of Slices : ' +
IntToStr(GetHeaderData(fs, vtLI, 4, 941, 944)));
Appendix – B, Database Schema

SET SQL DIALECT 3;

CREATE DATABASE 'nrl_svr\dev\hb\unisImageproc\data\IMAGEDB.GDB'
PAGE_SIZE 8192

DEFAULT CHARACTER SET

/* Table: CASES, Owner: SYSDBA */
CREATE TABLE "CASES"
(
  "CASEID" INTEGER NOT NULL,
  "PATIENTID" VARCHAR(50) NOT NULL,
  "CASETITLE" VARCHAR(50) NOT NULL,
  "STARTDATE" TIMESTAMP,
  "ENDDATE" TIMESTAMP,
  "CASETYPE" VARCHAR(50),
  CONSTRAINT "CASES_PK" PRIMARY KEY ("CASEID")
);

/* Table: IMAGEFILES, Owner: SYSDBA */
CREATE TABLE "IMAGEFILES"
(
  "FILEID" INTEGER NOT NULL,
  "FILENAME" VARCHAR(50),
  "FILEFULLNAME" VARCHAR(50),
  "IMAGE" BLOB SUB_TYPE 0 SEGMENT SIZE 80,
  "IMAGE_ORIG" BLOB SUB_TYPE 0 SEGMENT SIZE 80,
  "FILEHEADER" TEXT SEGMENT SIZE 80,
  "MAXINTENSITY" SMALLINT,
  "IMAGEWIDTH" SMALLINT,
  "IMAGEHEIGHT" SMALLINT,
  "IMAGEPIXELSIZE" SMALLINT,
  "ECHOTIME" SMALLINT,
  "FILESEQUENCE" VARCHAR(80),
  CONSTRAINT "IMAGEFILES_PK" PRIMARY KEY ("FILEID")
);

/* Table: IMAGEPARAMS, Owner: SYSDBA */
CREATE TABLE "IMAGEPARAMS"
(
  "IMAGEID" INTEGER NOT NULL,
  "PARAMID" INTEGER NOT NULL,
  "PARAMVALUE" VARCHAR(80),
  CONSTRAINT "IMAGEPARAMS_PK" PRIMARY KEY ("IMAGEID", "PARAMID")
);

/* Table: IMAGES, Owner: SYSDBA */
CREATE TABLE "IMAGES"
(
  "IMAGEID" INTEGER NOT NULL,
  "STUDYID" INTEGER NOT NULL,
  "IMAGENUMBER" INTEGER,
  "IMAGE" BLOB SUB_TYPE 0 SEGMENT SIZE 80,
  "MAXINTENSITY" SMALLINT,
  CONSTRAINT "IMAGES_PK" PRIMARY KEY ("IMAGEID")
);

/* Table: PATIENTS, Owner: SYSDBA */
CREATE TABLE "PATIENTS"
(
  "PATIENTID" INTEGER NOT NULL,
  "NAME" VARCHAR(100) NOT NULL,
  "BIRTHDATE" TIMESTAMP,
  "GENDER" VARCHAR(25),
  "NLCNUMBER" VARCHAR(80),
  "ADDRESS1" VARCHAR(80),
  "ADDRESS2" VARCHAR(80),
  "ADDRESS3" VARCHAR(80),
  "CITY" VARCHAR(50),
  "COUNTY" VARCHAR(50),
  "POSTCODE" VARCHAR(10),
  "COUNTRY" VARCHAR(10),
  "PHONE_HOME" VARCHAR(25),
  "PHONE_WORK" VARCHAR(25),
  "MOBILE" VARCHAR(25),
  "FAX" VARCHAR(25),
  CONSTRAINT "PATIENTS_PK" PRIMARY KEY ("PATIENTID")
);

/* Table: PLUGINS, Owner: SYSDBA */
CREATE TABLE "PLUGINS"
(
  "PLUGINID" INTEGER NOT NULL,
  "PLUGINTITLE" VARCHAR(35) NOT NULL,
  "PLUGINAPP" VARCHAR(80) NOT NULL,
  "PLUGINPARAM" VARCHAR(40),
  CONSTRAINT "PLUGINS_PK" PRIMARY KEY ("PLUGINID")
);
CREATE TABLE "STUDIES"  
(  "STUDYID" INTEGER NOT NULL,  
  "CASEID" INTEGER NOT NULL,  
  "STUDYNAME" VARCHAR(50),  
  "DESCRIPTION" VARCHAR(200) NOT NULL,  
  CONSTRAINT "STUDIES_PK" PRIMARY KEY ("STUDYID") 
);  
/* Table: T2SIMAGES, Owner: SYSDBA */  
CREATE TABLE "T2SIMAGES"  
(  "T2SIMAGEID" INTEGER NOT NULL,  
  "T2SIMAGE" BLOB SUB_TYPE 0 SEGMENT SIZE 80,  
  "MOIMAGE" BLOB SUB_TYPE 0 SEGMENT SIZE 80,  
  "T2SIMAGETITLE" VARCHAR(50) NOT NULL,  
  CONSTRAINT "T2SIMAGES_PK" PRIMARY KEY ("T2SIMAGEID") 
);  
/* Table: T2SIMAGESOURCES, Owner: SYSDBA */  
CREATE TABLE "T2SIMAGESOURCES"  
(  "T2SIMAGEID" INTEGER NOT NULL,  
  "FILEID" INTEGER NOT NULL,  
  CONSTRAINT "T2SIMAGESOURCES_PK" PRIMARY KEY ("T2SIMAGEID", "FILEID") 
);  
/* Table: TABLEINDEX, Owner: SYSDBA */  
CREATE TABLE "TABLEINDEX"  
(  "TABLENAME" VARCHAR(50) NOT NULL,  
  "LASTKEY" INTEGER NOT NULL,  
  CONSTRAINT "TABLEINDEX_PK" PRIMARY KEY ("TABLENAME") 
);  
/* Index definitions for all user tables */  
CREATE INDEX "CASES_SI_TITLE" ON  
"CASES"("CASETITLE");  
CREATE UNIQUE INDEX "IMAGEFILES_FN" ON  
"IMAGEFILES"("FILENAME");  
CREATE INDEX "IMAGEFILES__FS" ON  
"IMAGEFILES"("FILESEQUENCE");  
CREATE INDEX "IMAGES_SI_SID" ON  
"IMAGES"("STUDYID");  
CREATE INDEX "PARAMS_SI_NAME" ON  
"PARAMS"("PARAMNAME");  
CREATE UNIQUE INDEX "PATIENTS_SI_NAME" ON  
"PATIENTS"("NAME");  
CREATE UNIQUE INDEX "PLUGIN_SI_TITLE" ON  
"PLUGINS"("PLUGINTITLE");  
CREATE INDEX "STUDIES_SI_NAME" ON  
"STUDIES"("STUDYNAME");  
CREATE UNIQUE INDEX "T2SIMAGES_SI" ON  
"T2SIMAGES"("T2SIMAGETITLE");  
COMMIT WORK;  
SET AUTODDL OFF;  
CREATE PROCEDURE "GENNEWKEY"  
(  "TBLNAME" CHAR(30)  
)  
RETURNS  
(  "KEYVALUE" INTEGER  
)  
AS  
BEGIN  
/* update index table */  
Update TableIndex  
Set LastKey=LastKey+1  
Where upper(TableName)=upper(:tblName);  
/* get new key from Index table */  
Select LastKey  
From TableIndex  
Where upper(TableName)=upper(:tblName)  
Into :KeyValue;  
suspend;  
END  
/* Grant Roles for this database */  
/* Grant permissions for this database */  
*/  
/* END OF CODING */